HERRAMIENTA SOFTWARE COMO APOYO A LA METODOLOGÍA DE DISEÑO DE BASE DE DATOS RELACIONALES, QUE GENERE LAS INSTRUCCIONES NECESARIAS PARA LA CREACIÓN DE LA BASE DE DATOS

LEIDY CAROLINA CAMACHO HERNÁNDEZ
ANDRES FELIPE SOLANO ROMERO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

HERRAMIENTA SOFTWARE COMO APOYO A LA METODOLOGÍA DE DISEÑO DE BASE DE DATOS RELACIONALES, QUE GENERE LAS INSTRUCCIONES NECESARIAS PARA LA CREACIÓN DE LA BASE DE DATOS

LEIDY CAROLINA CAMACHO HERNÁNDEZ
ANDRES FELIPE SOLANO ROMERO

TESIS DE GRADO: TRABAJO DE INVESTIGACION PARA OPTAR AL TITULO DE INGENIERO DE SISTEMAS

DIRECTOR: CARLOS FELIPE REYES CONTRERAS
INGENIERO DE SISTEMAS DOCENTE CÁTEDRA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

AGRADECIMIENTOS

Queremos brindar nuestro más sincero agradecimiento a Dios por darnos la sabiduría suficiente para recorrer este camino; a nuestras familias por su amor y comprensión, porque siempre nos han brindado su apoyo y confianza y con sus esfuerzos permitieron este logro profesional; a nuestros amigos y compañeros de carrera por los grandes momentos compartidos y su valiosa colaboración; a la Escuela de Ingeniería de Sistemas e Informática por ser una guía constante a lo largo de estos años y finalmente a nuestro director de proyecto por su excelente labor, por su apoyo y su dedicación, mil y mil gracias porque este proyecto surgió y salió adelante gracias a sus ideas.

TABLA DE CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE TABLAS

Pág.

GLOSARIO

Atributo: Aquellas características, propiedades o cualidades de la entidad para las cuales es importante registrar su valor o cambio en ellas.

Base de Datos: Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada.

LDD: Un lenguaje de definición de datos que incluye comandos para crear la estructuras de la base de datos y sus elementos.

LMD: Un lenguaje de manipulación de datos que incluye comandos para insertar, actualizar, eliminar y recuperar datos dentro de la base de datos.

Entidad: Cualquier objeto, real o abstracto, que existe en un contexto determinado o puede llegar a existir y del cual deseamos guardar información.

Llave foránea: Columna o combinación de columnas cuyos valores se relacionan con la llave primaria de alguna otra tabla.

Metodología: Conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de un producto software.

Modelo de Datos Relacional: Modelo de datos lógico compuesto de relaciones, atributos y dominios.

Módulo: Unidad que forma parte de un sistema; en cuanto a su operación es independiente de otros módulos, pero en cuanto a su utilización puede requerir datos o información proporcionada por otros módulos del sistema que lo contiene.

Propietario del producto (Product Owner): Termino habitual para designar el rol de “responsable del producto”. Desempeñado por una persona conocedora del entorno del negocio del cliente y de la visión del producto y se responsabiliza por las fechas y funcionalidades de las diferentes versiones del producto.

Relación: Representan asociaciones entre entidades, se denominan el elemento del modelo que permite relacionar en sí los datos**.**

Scrum: Scrum es una metodología iterativa e incremental de gestión, mejora y mantenimiento de sistemas existentes o prototipos de producción.

Sentencia ALTER: Permite variar la definición de una tabla ya creada, admite modificar la definición de columnas, añadir más columnas o restricciones, eliminar columnas y restricciones y habilitar/deshabilitar restricciones.

Sentencia Create: Es la sentencia utilizada para crear y definir un objeto en la base de datos.

Sentencia Drop: Este comando es utilizado para eliminar un objeto de la base de datos.

Sistema de Gestión de Base de Datos (SGBD): Conjunto coordinado de programas, procedimientos, lenguajes, herramientas, etc., que suministra los medios necesarios para describir, actualizar y administrar los datos de una BD.

Sprint (Iteración): Ciclo de tiempo en el que se desarrolla cada incremento iterativo del producto. La duración habitual de cada sprint no suele ser inferior a una semana, ni superior a un mes.

SQL: Lenguaje de consulta estructurado (Structured Query Languaje), que nos sirve para definir y manipular los datos de una base de datos relacional.

Tipo de dato: Identificador que especifica la clase de información (numeros, valores logicos, caracteres, texto, etc) que contiene una columna y como será almacenada.

Universo del Discurso (UD): Parte del mundo real que queremos almacenar en una base de datos.

RESUMEN

Título: HERRAMIENTA SOFTWARE COMO APOYO A LA METODOLOGÍA DE DISEÑO DE BASE DE DATOS RELACIONALES, QUE GENERE LAS INSTRUCCIONES NECESARIAS PARA LA CREACIÓN DE LA BASE DE DATOS*

AUTOR: Leidy Carolina Camacho Hernández / Andrés Felipe Solano Romero **

PALABRAS CLAVE: Bases de datos relacionales MySQL PostgreSQL Scrum metodología

DESCRIPCIÓN:

En este documento se describe el proceso desarrollado por los autores del proyecto. Este producto y su documentación se reflejaron en una serie de capítulos que serán resumidos a continuación:

Marco Teórico
Esta sección describe la investigación previa que fue indispensable realizar para lograr el objetivo del proyecto. Se incluyen conceptos básicos que explican las bases de datos, su historia y funcionalidad, adicionalmente los métodos de modelado y diseño de bases de datos.

Desarrollo e Implementación de la Metodología del Proyecto
Esta sección describe la explicación de la metodología Scrum utilizada para el desarrollo del proyecto, el diseño de las interfaces de usuario, los módulos que conforman la herramienta y los resultados obtenidos luego de integrar los módulos.

El Producto y su Documentación
Esta sección presenta la explicación de la metodología de diseño de bases de datos relacionales aplicada en la herramienta software y un recorrido por el sistema que describe paso a paso el uso de la herramienta mediante ayudas gráficas y la explicación detallada tanto del producto como de los resultados generados luego de la culminación de la implementación.

El lenguaje SQL y las sentencias para la creación de la base de datos
Esta sección presenta la definición y la historia de SQL, además de la explicación de las sentencias usadas para la creación de la base de datos.

* Trabajo de Grado

** Facultad De Ingenierías Fisicomecánicas. Escuela Ingeniería De Sistemas E Informática. Director: Carlos Felipe Reyes Contreras

SUMMARY

Title: SOFTWARE TOOL TO SUPPORT A DESIGN METHODOLOGY OF RELATIONAL DATABASE, WHICH GENERATES INSTRUCTIONS NEEDED TO CREATE DATABASE.*

AUTHOR: Leidy Carolina Camacho Hernández / Andrés Felipe Solano Romero **

KEY WORDS: Relational Databases MySQL PostgreSQL Scrum methodology

CONTENT:

This paper describes the process developed by the authors of the project. This product and its documentation are reflected in a series of chapters which will be summarized below:

Theoretical Framework

This section and its content describes the previous research that was indispensable to achieve the project objective. There is included basic concepts that explain databases, their history and functionality, in addition the methods of modeling and design of databases.

Development and Implementation of Project Methodology

This section and its content describes the explanation of the Scrum methodology used to develop the project, designing user interfaces, modules as part of the tool and the results obtained after integrating modules.

The Product and Documentation

This section ant its content provides an explanation of the methodology for designing relational databases applied to the software tool and a tour of the system that describes step by step the use of the tool with graphic aids and detailed explanation of both, the product, and the results generated after the completion of the implementation.

The SQL language and statements to create the database

This section and its content presents the definition and history of SQL, in addition, the explanation of the statements used to create the database.

* Work Degree

** Faculty of Engineering physicomechanical. School E Computer Systems Engineering. Director: Carlos Felipe Reyes Contreras

INTRODUCCION

La gestión de las bases de datos ha evolucionado desde una aplicación informática especializada hasta una parte esencial de un entorno informático moderno, conllevando al desarrollo de un amplio conjunto de conceptos y técnicas para la gestión de los datos.

Para lograr usar adecuadamente la tecnología de las bases de datos relacionales, es necesario complementar estos conocimientos con un aspecto fundamental, un buen diseño del modelo de la base de datos, para el cual es necesario establecer un proceso partiendo del mundo real, de tal forma que sea posible plasmarlo a partir de una serie de objetos básicos llamados entidades y de relaciones entre estos objetos.

El desarrollo de software de este proyecto propone una herramienta informática que pretende trabajar dentro del contexto de problemas cotidianos y le permite a usuarios con conocimientos mínimos, a través de una serie de pasos didácticos, diseñar fácilmente un modelo de base de datos, adecuarlo a sus necesidades, y obtener como resultado, las sentencias de creación de la base de datos.

GBD es un sistema totalmente instructivo para el diseño del modelo de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecer un método efectivo para gestionar bases de datos sin necesidad de programar.

Por último, el proceso desarrollado durante el proyecto se documentó en este libro, en él se explica en que consiste una base de datos, su funcionalidad y algunos conocimientos básicos sobre el tema, la realización de la estructura funcional partiendo desde la percepción del usuario, los módulos que conforman la

herramienta y que integrados permitan generar las sentencias de creación de la base de datos en MySQL y PostgreSQL y finalmente se documentará el método que utiliza la herramienta y un breve manual de uso. Es importante resaltar que el proyecto se desarrolló bajo la metodología Scrum.

1. DESCRIPCIÓN DEL PROYECTO

1.1. DESCRIPCIÓN DE LA PROBLEMÁTICA

Muchos de los desarrollos de software requieren almacenar y gestionar la información almacenada en una base de datos. El diseño de un modelo de base de datos aceptablemente consistente no es una tarea sencilla. En algunos casos puede convertirse en una labor desgastante e imprecisa, que puede conllevar a ajustes del software sobre la marcha. Existen diversas metodologías de diseño de base de datos, algunas demasiado complejas. La metodología de diseño de bases de datos relacionales resulta bastante didáctica y fácil de aplicar; sin pretender con esto afirmar que su resultado será el más preciso, sin embargo, la aplicación del paso a paso que ofrece esta herramienta podrá garantizar que se obtenga un modelo bastante aceptable.

Existen en la actualidad muchas herramientas buenas y excelentes; que ayudan a generar las sentencias de creación de la base de datos con muy buenos resultados, pero requieren que se tenga un modelo ya definido.

El software que se propone construir es una herramienta de apoyo a la aplicación de la metodología de bases de datos relacionales; que ofrece un asistente muy didáctico e intuitivo y que permite que cualquier persona con los conocimientos mínimos, pueda diseñar un modelo final de la base de datos lo suficientemente aceptable.

1.2. OBJETIVOS

A continuación se enuncian de manera organizada y explicita los propósitos del proyecto que conllevaron al esfuerzo de los autores durante todo el desarrollo del proyecto:

1.2.1. Objetivo General

Implementar una herramienta software como complemento a la metodología de diseño de base de datos relacionales; que facilite la aplicación paso a paso de la metodología y genere como resultado las instrucciones SQL para la creación de la base datos.

1.2.2. Específicos

- Recopilar información relacionada con bases de datos relacionales, documentando los conceptos y terminología de mayor incidencia para el desarrollo del proyecto.

- Diseñar las interfaces de usuario que constituirán la herramienta software.

- Implementar un sistema que facilite la ejecución paso a paso de la metodología propuesta.

- Generar las sentencias SQL requeridas para la creación de la base de datos.

- Elaborar la documentación respectiva, que le facilite a los usuarios finales la usabilidad de la herramienta software.

2. MARCO TEÓRICO

2.1. FUNDAMENTOS TEÓRICOS

El éxito del desarrollo de una base de datos depende principalmente del dominio y entendimiento de los conceptos fundamentales que conforman una base de datos. La herramienta software que se desarrolló en este proyecto se cimentó en la teoría expuesta a continuación y la metodología SCRUM.

2.1.1. Historia de las bases de datos. Los orígenes de las bases de datos y sus primeros usos se remontan a la antigüedad donde ya existían bibliotecas y toda clase de registros. También se utilizaban para recoger información sobre las cosechas y realizar censos. Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual que resultaba muy tedioso. El gran desarrollo de las bases de datos surgió principalmente a partir de las necesidades de almacenar grandes cantidades de información o datos, para su posterior consulta, producidas en su mayoría por las nuevas industrias que generaban mucha información.

Alrededor de 1880 los censos se realizaban de forma manual. El censo de Estados Unidos tardo 7 años para obtener resultados, pero Herman Hollerith en 1884 creó la máquina perforadora, la cual, produjo resultados en solo dos años y medio en el censo de 1890 donde era posible obtener datos importantes como el número de nacimientos, población infantil y número de familias siendo nombrado así el primer ingeniero estadístico de la historia.

En la década de los cincuentas se desarrollaron las cintas magnéticas para poder almacenar datos, haciendo posible la automatización de procesos tales como las

nóminas. Consistía en leer datos de una o varias cintas y pasarlos a otra, al igual se podían pasar desde las tarjetas perforadas simulando un back-up. Estas cintas al igual que los paquetes de tarjetas perforadas solo se podían leer secuencial y ordenadamente requiriendo altas cantidades de tiempo para realizar las operaciones con ellas.

Posteriormente en la época de los sesenta, se popularizó el uso de los discos como consecuencia de los bajos precios de los computadores, convirtiéndose en un adelanto muy efectivo en la época, debido a que permitía consultar la información directamente sin tener que saber la ubicación exacta de los datos.

Por lo que respecta a la década de los setenta, Edgar Frank Codd definió el modelo relacional, al mismo tiempo publicó una serie de reglas para los sistemas de datos relacionales a través de su artículo “Un modelo relacional de datos para grandes bancos de datos compartidos”. En 1976 aparece un modelo de base de datos nuevo llamado Entidad-Relación permitiendo a los diseñadores centrarse en la aplicación de los datos en lugar de la estructura de la tabla lógica.

Por su parte, a principios de los años ochenta las bases de datos relacionales supusieron un avance importante para facilitar la programación, realizando automáticamente casi todas las tareas de bajo nivel, liberando al programador en el nivel lógico. A su vez, el modelo relación consiguió el reinado supremo entre todos los modelos de datos. También en ese tiempo, se dio una gran investigación en las bases de datos paralelas y distribuidas, así como el trabajo inicial en las bases de datos orientadas a objetos.

El principal acontecimiento a finales de los 90’s fue el crecimiento explosivo de World Wide Web. Las bases de datos entonces se implantaron mucho más extensivamente que nunca antes. Los sistemas de bases de datos ahora soportan tasas de transacciones muy altas, así como alta fiabilidad y disponibilidad,

eliminando los tiempos planificados de inactividad por mantenimiento. También tuvieron interfaces Web a los datos, pues facilitaba su consulta.

Hoy en día, los sistemas de base de datos relacionales están en plena transformación para adaptarse a tres tecnologías de éxito reciente: la multimedia, la orientación a objetos y la Web. La rápida adopción de la Web a los sistemas de información hace que los sistemas de base de datos incorporen recursos para ser servidores de páginas Web, como por ejemplo la inclusión de SQL en guiones HTML, SQL incorporado en Java, etc.

En los últimos años se ha empezado a extender un tipo de aplicación de bases de datos denominado Data Warehouse, o almacén de datos, que produce cambios en los sistemas de bases de datos relacionales del mercado. Las empresas durante muchos años han trabajado con bases de datos, acumulando gran cantidad de datos de todo tipo los cuales si se analizan apropiadamente pueden dar información valiosa. Los datos de este gran almacén, el Data Warehouse, se obtienen por una replicación más o menos elaborada de las que hay en las bases de datos que se utilizan en el trabajo cotidiano de la empresa.

2.1.2. Dato. No es sencillo definir qué es un dato, pero intentaremos conceptualizarlo desde el punto de vista de las bases de datos. Podemos decir que [1]un dato es todo elemento de información que necesita una organización para su funcionamiento, es decir es toda información que refleja el valor de una característica de un objeto real, sea concreto o abstracto, o imaginario, que debe cumplir algunas condiciones, además, debe tener un significado y debe ser manipulable mediante operadores.

[1] Olga Pons, Nicolás Marín, Juan Miguel Medina, Silvia Acid & Mª Amparo Vila: “Introducción a las Bases de Datos: El modelo relacional”. Paraninfo, 2005. ISBN 8497323963. Thomson Editores

Los datos también se definen como expresiones generales que describen características de las entidades sobre las que operan los algoritmos. Estas expresiones deben presentarse de una cierta manera para que puedan ser tratadas por una computadora, lo que implica que los datos por sí solos no poseen relevancia, tampoco constituyen información, sino que ésta surge del adecuado procesamiento de los datos y así puede llevarnos a una conclusión.

2.1.3. Información. Según Idalberto Chiavenato la información consiste en un conjunto de datos que poseen un significado, de modo tal que reducen la incertidumbre y aumentan el conocimiento de quien se acerca a contemplarlos. Estos datos se encuentran disponibles para su uso inmediato y sirven para clarificar incertidumbres sobre determinados temas.

La información está constituida por un grupo de datos ya supervisados y ordenados, que sirven para construir un mensaje basado en un cierto fenómeno o ente, al mismo tiempo, la información permite resolver problemas y tomar decisiones, ya que su aprovechamiento racional es la base del conocimiento, por lo tanto, la información es un recurso que otorga significado o sentido a la realidad, ya que mediante códigos y conjuntos de datos, da origen a los modelos de pensamiento humano.

Otros autores como Czinkota y Kotabe han definido que la información consiste en un conjunto de datos que han sido clasificados y ordenados con un propósito determinado.

Figura 1. Pirámide de valor

Fuente: FRASSIA, Mercedes. Introducción a las bases de datos, un poco de historia. Argentina. 2001. 34 p.

2.1.4. Bases de datos. La palabra datos hace referencia a los datos conocidos que se pueden grabar y que tienen un significado implícito. Esta colección de datos relacionados con un significado implícito es una base de datos. En sí, es un conjunto de datos relacionados y almacenados estructuradamente para su posterior uso. También se puede definir como un “almacén” que permite guardar grandes cantidades de información de forma organizada para ser empleada fácilmente.

Una base de datos representa algún aspecto del mundo real, lo que en ocasiones se denomina mini-mundo o universo de discurso. Los cambios introducidos en el mini-mundo se reflejan en la base de datos. En otras palabras, una base de datos tiene algún origen del que se derivan los datos, algún grado de interacción con eventos del mundo real y un público que está activamente interesado en su contenido. Los usuarios finales de una base de datos pueden producir eventos que provoquen un cambio en la información almacenada en la base de datos. Al objeto de que una base de datos sea en todo momento precisa y fiable, debe ser

un reflejo exacto del mini-mundo que representa; por consiguiente, en la base de datos deben reflejarse los cambios tan pronto como sea posible.

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

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Acceso a través de lenguajes de programación estándar.

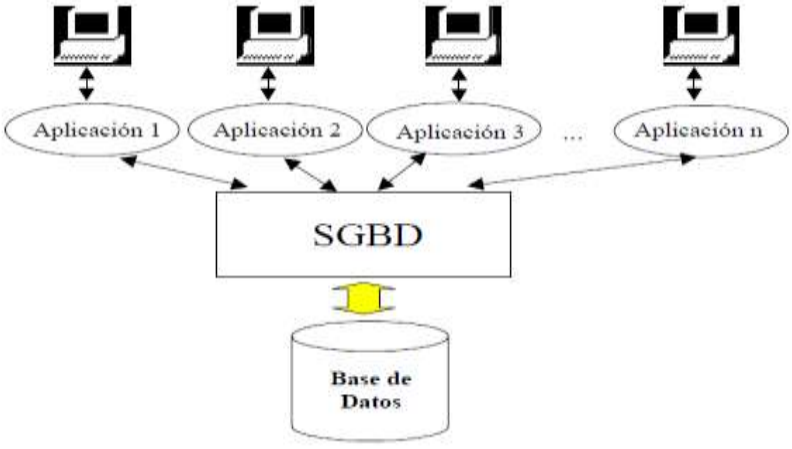
La principal ventaja de utilizar bases de datos radica en que múltiples usuarios pueden acceder a ella al mismo tiempo, pueden visualizar, ingresar o actualizar los datos, en concordancia con los derechos de acceso que se les hayan otorgado. Entre mayor sea la cantidad de datos que contenga, mayor valor toma su uso, debido a que le proporciona a los usuarios gran cantidad de información ordenada y con la mínima redundancia posible.

Una base de datos puede ser local, es decir que puede utilizarla sólo un usuario en un equipo, o puede ser distribuida, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red. La principal ventaja de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo.

2.1.5. Sistema de Gestión de Base de Datos (SGBD). Después de la creación de las bases de datos nace la inquietud sobre cómo controlar gran cantidad de datos como el número de usuarios. Dicha administración se realiza con un sistema llamado (SGBD) Sistema de Gestión de Base de Datos, los cuales son un tipo de

herramienta software dedicados específicamente a servir como interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan y consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos.

Figura 2. Sistema Gestor de Bases de Datos



Fuente: CUADRA, Dolores; CASTRO, Elena y MARTÍNEZ, Paloma. Desarrollo de Bases de Datos: casos prácticos desde el análisis a la implementación. España: RA-MA S.A. Editorial y Publicaciones, 2007. 569 p.

Estos están compuestos por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta.

El sistema de administración de bases de datos, puede dividirse en tres subsistemas:

- El sistema de administración de archivos: permite almacenar la información en un medio físico.
- El SGBD interno: permite ordenar la información.
- El SGBD externo: representa la interfaz del usuario.

El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información, la gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información, además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización.

2.1.6. Objetivos de los Sistemas De Gestión De Base De Datos (SGBD). Existe un conjunto de normas que deben cumplir los SGBD para facilitar el diseño y manipulación de aplicaciones, que les permitan a los usuarios interactuar con las bases de datos con mayor flexibilidad. Estos objetivos son:

Independencia de los datos y los programas de aplicación: La independencia de los datos se define como la inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso, constituyéndose como el objetivo fundamental de los SGBD.

En un SBD sería indeseable la existencia de aplicaciones y datos dependientes entre sí, por dos razones fundamentales:

- Diferentes aplicaciones necesitarán diferentes aspectos de los mismos datos.
- Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios hechos en la realidad sin necesidad de modificar los programas de aplicación.

Minimización de la redundancia: Uno de los objetivos de los SGBD es minimizar la redundancia de los datos, lo que se trata de lograr es disminuir la redundancia

innecesaria, para disminuir el tiempo de acceso a los datos o para simplificar el método de direccionamiento.

Integración y sincronización de las bases de datos: La sincronización consiste en la necesidad de garantizar el acceso múltiple y simultáneo a la base de datos, permitiendo que los datos puedan ser compartidos por diferentes usuarios a la vez. La integración consiste en garantizar una respuesta a los requerimientos de diferentes aspectos de los datos por diferentes usuarios, logrando entregarle al programa de aplicación los datos que solicita y en la forma en que los solicita.

Integridad de los datos: Consiste en garantizar que no haya incoherencia en los datos almacenados, de modo que los sean correctos en cualquier momento. Está directamente relacionada con la exactitud de la información contenida y la minimización de la redundancia.

Seguridad y recuperación: [2]La seguridad garantiza el acceso autorizado a los datos, la forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención. La recuperación permite que el sistema de bases de datos disponga de métodos que garanticen la restauración de las bases de datos al producirse alguna falla técnica, interrupción de la energía eléctrica, etc.

Facilidad de manipulación de la información: El SGBD debe contar con la capacidad de una búsqueda rápida por diferentes criterios, debe permitir que los usuarios planteen sus requisitos de información de una forma simple.

[2] Bases de datos teoría. Disponible en Internet: https://cibertec.googlecode.com/files/Base%20de%20Datos%20Teor%C3%ADa.pdf [Consultado 28 de Enero de 2014].

Control centralizado: Debe permitirle a una persona (administrador de la base de datos) o conjunto de personas, controlar de manera sistemática y única, los datos que se almacenan en la base de datos, así como el acceso a ella.

2.1.7. Lenguajes de bases de datos. Un sistema de bases de datos proporciona un lenguaje de definición de datos para especificar el esquema de la base de datos y un lenguaje de manipulación de datos para expresar las consultas a la base de datos y las modificaciones.

2.1.7.1 Lenguaje de definición de datos. Un esquema de base de datos se especifica utilizando un conjunto de definiciones expresadas mediante un lenguaje especial llamado lenguaje de definición de datos (LDD).

El LDD es la parte dedicada a la definición de la base de datos que permite especificar gran parte del nivel interno de la base de datos y consta de sentencias que permiten especificar la estructura de la base de datos, por lo que se induce que estas sentencias serán utilizadas normalmente por el administrador de la base de datos.

La definición de la estructura de la base de datos incluye tanto la creación inicial de los diferentes elementos que formarán la base de datos, como el mantenimiento de esa estructura. Las sentencias del LDD utilizan unos verbos que se repiten para los distintos objetos. Por ejemplo para crear un objeto nuevo el verbo será CREATE y a continuación el tipo de objeto a crear. CREATE DATABASE es la sentencia para crear una base de datos, CREATE TABLE nos permite crear una nueva tabla, CREATE INDEX crear un nuevo índice… Para eliminar un objeto utilizaremos el verbo DROP (DROP TABLE, DROP INDEX…) y para modificar algo de la definición de un objeto ya creado utilizamos el verbo ALTER (ALTER TABLE, ALTER INDEX…).

2.1.7.2 Lenguaje de manipulación de datos. Un lenguaje de manipulación de datos (LMD) es un lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios llevar a cabo las tareas de consulta o modificación de los datos contenidos en las Bases de Datos del Sistema Gestor de Bases de Datos. La manipulación de datos permite básicamente la recuperación de información almacenada en la base de datos, la inserción de información nueva en la base de datos, el borrado de información de la base de datos y la modificación de información almacenada en la base de datos, entre otras.

El LMD está formado por cuatro sentencias, que proporcionan el acceso al contenido de la base de datos y no modifican la estructura de la misma, estas son: la sentencia SELECT, consulta el contenido de la base de datos; la sentencia INSERT, inserta filas dentro de tablas de la base de datos; la sentencia UPDATE, modifica el contenido de las tablas en la base de datos y la sentencia DELETE, borra filas dentro de las tablas de la base de datos.

2.2. MÉTODOS DE DISEÑO Y MODELADO DE BASES DE DATOS

El diseño de bases de datos se ha convertido en una actividad crucial en el ambiente de bases de datos, ya que permite el uso de los datos desde perspectivas más complejos. Las clases de estructuras de datos creadas dentro de la base de datos y las relaciones entre ellas desempeñan una poderosa función cuando se utilizan modelos.

Los modelos se definen como abstracciones simplificadas de eventos y condiciones del mundo real, lo que implica que si los modelos son lógicamente buenos, se obtendrán diseños de bases de datos funcionales que permitan obtener información realmente útil. La búsqueda de una mejor administración de

datos conllevo a la representación de los datos por diversos modelos de bases de datos.

Un modelo de datos se define como una representación abstracta de los datos y las relaciones entre ellos, es una colección de herramientas conceptuales para la descripción de datos, relaciones entre datos, semántica de los datos y restricciones de consistencia. Los modelos de datos son esenciales para el desarrollo de sistemas de información, ya que a través de ellos puede conseguirse la compatibilidad necesaria para manejar gigantescas cantidades de datos. Además estos permiten describir las estructuras de datos de la base (el tipo de los datos que hay en la base de datos y la forma en que se relacionan), las restricciones de integridad (conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (operaciones de agregado, borrado, modificación y recuperación de los datos de la base de datos).

Estos modelos de datos se pueden agrupar en dos categorías: modelos conceptuales y modelos de ejecución

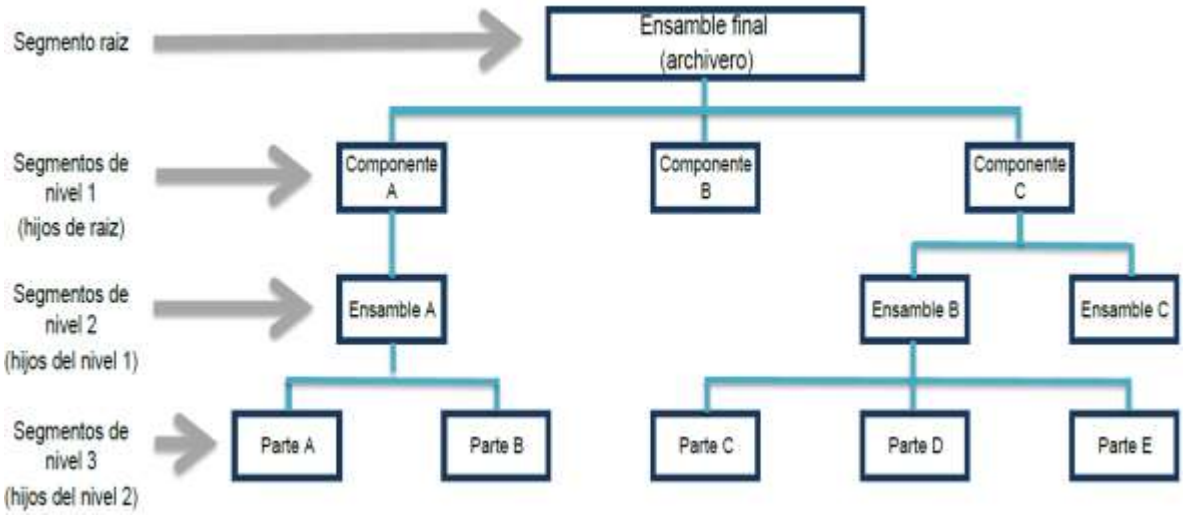
- El modelo conceptual se orienta en representar la naturaleza lógica de los datos, por lo tanto, está comprometido con lo que está representado en la bases de datos y no en cómo está representado. Los modelos conceptuales incluyen el modelo Entidad - Relación y el modelo orientado a objetos.
- [3]El modelo de ejecución hace énfasis en cómo los datos están representados en la base de datos o en cómo se ejecutan las estructuras de datos para representar lo que está modelado. Los modelos de ejecución incluyen el modelo de bases de datos jerárquico, el de bases de datos de red, el modelo

[3] SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN. “Fundamentos de bases de datos”. Cuarta edición. Madrid: McGraw-Hill, 2002.

de bases de datos relacional y el modelo de bases de datos orientado a objetos.

2.2.1. Modelo jerárquico. La base de datos jerárquica es un conjunto de registros lógicamente organizados de conformidad con la estructura de un árbol invertido en donde el usuario percibe la base de datos como una jerarquía de segmentos. Un segmento es el equivalente a un tipo de registro de sistema. El nivel superior (la raíz) se percibe como el padre de los segmentos ubicados directamente debajo de él (nivel 1), que a su vez se convierten en los padres de los segmentos del nivel 2, y así sucesivamente; a su vez los segmentos debajo de otros son los hijos del que está arriba de ellos.

Figura 3. Modelo Jerárquico



Fuente: ROB, Peter; CORONEL, Carlos. “Sistemas de bases de datos: diseño, implementación y administración”. Quinta edición. México: International Thomson Editores, 2004

Por lo tanto, es posible afirmar que es fácil explorar tanto los componentes de la base de datos como las relaciones uno a muchos entre ellos, dado que cada padre puede tener muchos hijos, pero cada hijo solo puede tener un padre, el

problema es que el computador no ve la estructura lógica del árbol, como se ve en físico, para ello se define una ruta que rastrea los segmentos padres hacia los segmentos hijos, iniciando por la izquierda, ordenando secuencialmente los segmento a través de una ruta jerárquica la cual considera que los segmentos accesados con mayor frecuencia deben localizarse lo más cerca posible del lado izquierdo del árbol, para garantizar un acceso más eficiente a los datos, de la siguiente forma:

Ensamble final → Componente A → Ensamble A → Parte A → Parte B → Componente B → Componente C → Ensamble B → Parte C → Parte D

El modelo jerárquico es más efectivo, siempre y cuando se tengan muchas transacciones que impliquen relaciones uno a muchos que permanezcan siempre fijas, por ello el modelo ofreció muchas ventajas sobresalientes como por ejemplo:

- Independencia de los datos: se disminuye el esfuerzo de programación y mantenimiento del programa dado que su estructura en cascada facilita cualquier tipo de cambio en el tipo de dato, con lo que se elimina la necesidad de realizar cambios en los segmentos del programa que hacen referencia al tipo de datos que fue alterado.
- Simplicidad conceptual: el proceso de diseño se ve simplificado dado que en su estructura jerárquica, la relación entre los niveles del árbol es lógicamente simple, lo que ayuda a comprender conceptualmente la base de datos.
- Eficiencia: Si el modelo de con el que se diseña la base de datos permite grandes cantidades de datos con relaciones uno a muchos y que dichas relaciones y transacciones se mantengan fijas con el tiempo, se afirma que este modelo es altamente eficiente.

- Integridad: Dadas las relaciones automáticas entre padre e hijo, el vínculo directo entre segmento padre y sus hijos fomenta la integridad de la base de datos.
- Seguridad: Dado que el modelo está diseñado uniformemente, la seguridad se ejecuta igualmente, evitando esfuerzos adicionales por parte de los programadores que impliquen aplicaciones individuales con respecto a los diversos grados o tipos de datos que requieran seguridad adicional.

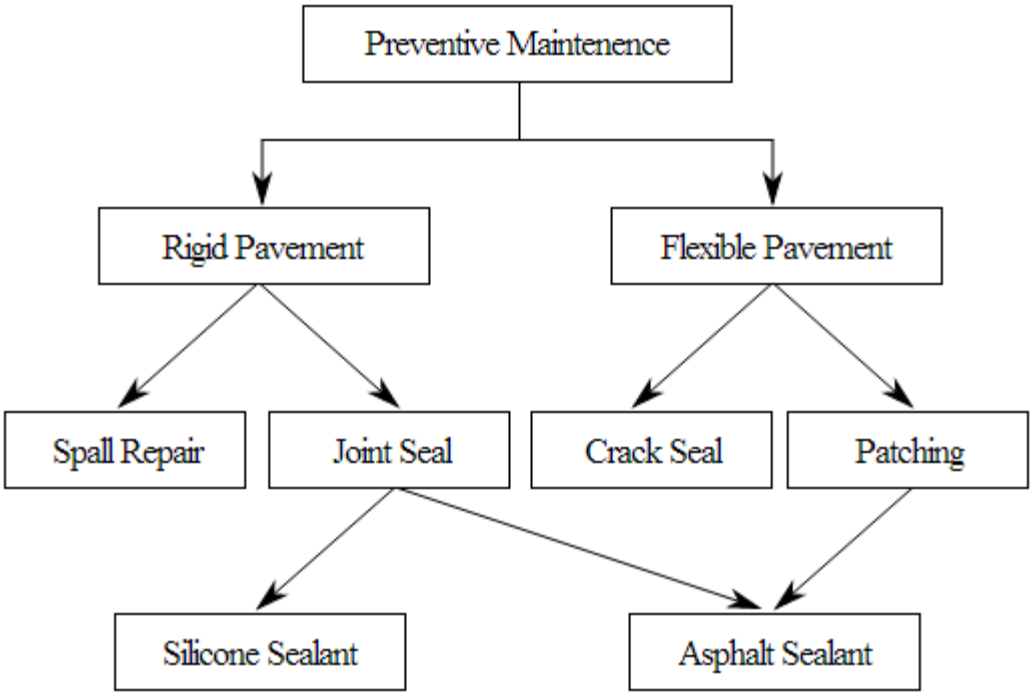
El modelo jerárquico cayó a principios de los años 80 dado a sus grandes inconvenientes; el modelo de base de datos de red fue el más sobresaliente entre varios modelos alternos que se desarrollaron en la época.

2.2.2. Modelo de base de datos de red. El modelo de bases de datos de red tiene algunas similitudes con el modelo jerárquico. Los dos modelos permiten que el usuario perciba la base de datos como un conjunto de registros con relaciones uno a muchos pero adicionalmente en el modelo de red un registro puede tener más de un padre.

[4]En la terminología de bases de datos de red, una relación se llama conjunto. Cada conjunto se compone de por lo menos dos tipos de registro: un registro propietario que equivale al padre del modelo jerárquico y un registro miembro que equivale al hijo del modelo jerárquico. La diferencia entre el modelo jerárquico y el de red es que este podría incluir una condición en la que un registro puede aparecer como miembro en más de un conjunto. Un conjunto representa una relación uno a muchos entre el propietario y el miembro, lo que indica que un miembro puede tener varios propietarios.

[4] ROB, Peter y CORONEL, Carlos. Sistemas de bases de datos: diseño, implementación y administración. Quinta edición. México: International Thomson Editores, 2004. 838 p.

Figura 4. Modelo de Red



Fuente: Modelo de Red. Disponible en Internet: http://es.wikipedia.org/wiki/Modelo_de_red [Consultado 07 de Febrero de 2014].

El modelo de bases de datos de red conserva algunas ventajas del modelo jerárquico y al mismo tiempo corrige muchos de sus defectos como por ejemplo:

- Cumplimiento de estándares: El modelo de red se adapta a los estándares de base de datos impuestos en los años 70 e incluye un lenguaje de definición de datos (LDD) y un lenguaje de manipulación de datos (LMD) que facilita en gran medida la administración y portabilidad de las bases de datos.
- Maneja más tipos de relaciones: En este modelo es posible ejecutarse con mayor facilidad las relaciones muchos a muchos, lo que antes era casi imposible.
- Independencia de los datos: El modelo de red permite aislar parcialmente los programas de los detalles de almacenamiento físico lo que implica que los

cambios en las características de los datos no requieran cambios en las partes de acceso a los datos de los programas de aplicación.

- Flexibilidad de acceso a los datos: la flexibilidad de acceso a los datos y a los sistemas de archivos es altamente sobresaliente con respecto al método jerárquico, dado que en este método es posible que dentro de una aplicación se pueda tener acceso a un registro propietario y a todos los miembros dentro del conjunto, lo que indica que ya no se requiere una ruta preordenada para acceder a todos los registros de la base de datos.

2.2.3. Modelo de base de datos orientado a objetos. El primer modelo desarrollado de este tipo fue el modelo de datos semántico (SDM) que modelo tanto datos como sus relaciones en una sola estructura denominada objeto. Debido a su estructura básica de modelo es un objeto, se dice que el SDM es un modelo de datos orientado a objetos, el cual se convierte en la base del modelo de base de datos orientado a objetos, que es manejado a partir de un sistema de administración de base de datos orientado a objetos.

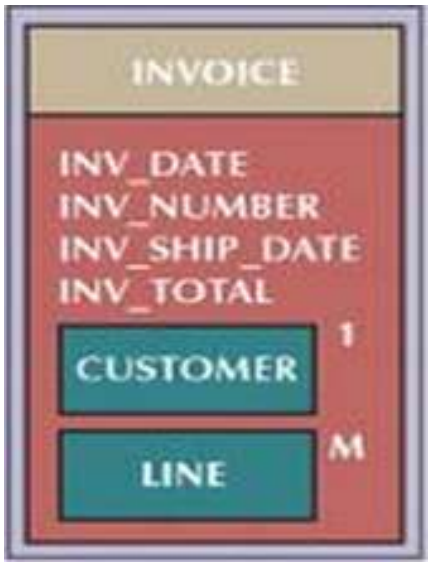
El modelo de datos orientado a objetos muestra que al igual que la entidad en un modelo de base de datos relacional, un objeto es descrito por su contenido de hechos, pero además de esto incluye información sobre la relación entre los hechos dentro del objeto, igual que información sobre sus relaciones con otros objetos; por lo tanto, a los hechos dentro del objeto se les otorga un mayor significado, además permite que un objeto incluya todas las operaciones que pueden ser realizadas en él.

El modelo de datos orientado a objetos está basado en los siguientes componentes:

- Los objetos del modelo de datos son abstracciones de entidades o eventos del mundo real.

- Los atributos describen las propiedades de un objeto.
- Los objetos que comparten características similares se agrupan en clases. Una clase es un conjunto de objetos similares con estructura (atributos) y comportamiento (métodos) compartidos. Un método representa una acción del mundo real, es decir, los métodos equivalen a los procedimientos en lenguaje de programación tradicionales.
- Las clases se organizan en una jerarquía de clase, dicha jerarquía simula un árbol invertido donde cada clase tiene un único padre.
- La herencia es la capacidad de un objeto dentro de la jerarquía de clase, de heredar los atributos y métodos de las clases que están sobre él.
- El modelo de datos orientado a objetos representa un objeto como un cuadro vertical: todos los atributos y las relaciones del objeto con otros objetos se incluyen dentro del cuadro objeto.

Figura 5. Representación de un objeto



Fuente: CORONEL, Carlos; MORRIS, Steven y ROB, Peter. Bases de Datos, Diseño, Implementación y Administración. Novena edición. México: Cengage Learning Editores, 2011.250 p.

El modelo de datos orientado a objetos ofrece algunas ventajas como por ejemplo:

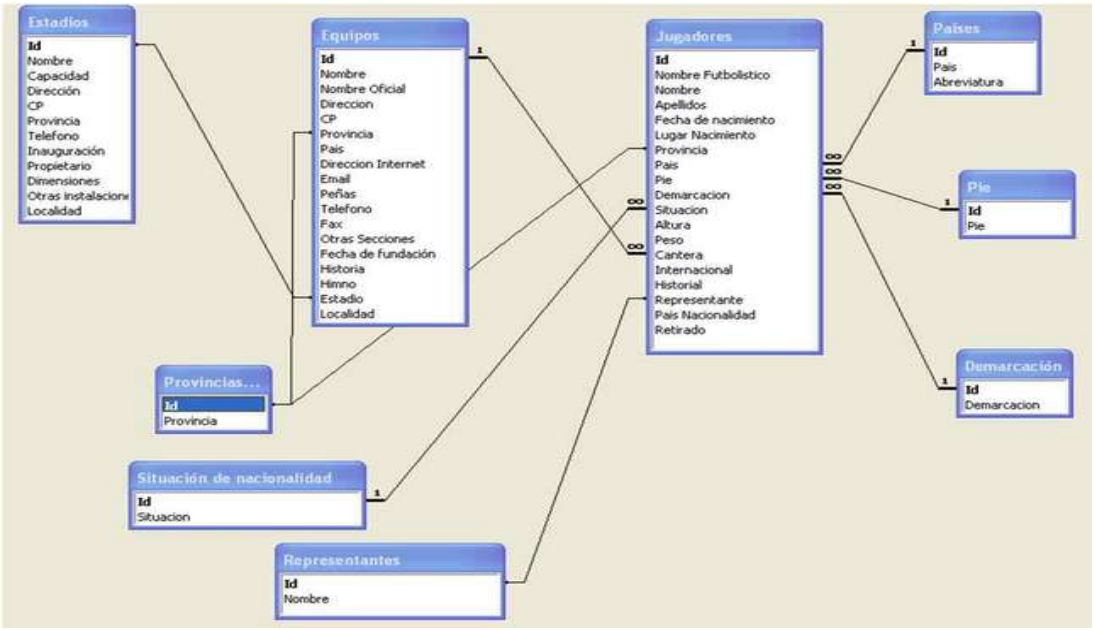
- La presentación visual incluye contenido semántico ya que modela visualmente las relaciones, pero adicionalmente incluye contenido semántico dentro de la representación visual del objeto, lo que facilita visualizar relaciones mucho más complejas dentro y entre los objetos.
- Integridad de la base de datos: el modelo utiliza la herencia para proteger la integridad de la base de datos, pero adicionalmente incluye más tipos de relaciones, permitiendo mayor complejidad en ellas.
- Agrega contenido semántico: la adición del contenido semántico al modelo, le da mayor significado a los datos.
- Independencia estructural de los datos: la autonomía de los objetos del modelo de datos orientado a objetos garantiza la independencia estructural igual que la independencia de los datos.

2.2.4. Modelo Relacional. A pesar de los puntos a favor con que contaba el modelo de red, su complejidad estructural impidió que los usuarios y diseñadores pudieran sacarle el provecho suficiente, lo que conllevo a la aparición del modelo de base de datos relacional el cual se estableció como el principal modelo de datos para las aplicaciones de procesamiento de datos. Consiguió una posición primordial debido a su simplicidad, que facilitó el trabajo del programador en comparación con otros modelos anteriores como el de red y el jerárquico y además represento un avance sensacional y preparo el camino para una genuina revolución en el campo de las bases de datos.

Una base de datos relacional consiste en un conjunto de tablas, a cada una de las cuales se le asigna un nombre exclusivo. Cada tabla tiene una estructura donde se representaron las bases de datos mediante tablas, en donde cada fila de la tabla representa una relación entre un conjunto de valores.

Dado que cada tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación, del que toma su nombre el modelo de datos relacional.

Figura 6. Modelo Relacional



Fuente: ESCUDERO, Juan Manuel. Comprendiendo el Modelo Relacional. Disponible en internet: blog.iedge.eu/tecnologia-sistemas-informacion/bases-de-datos/juan-manuel-escudero-comprendiendo-el-modelo-relacional/. [Consultado 15 de Febrero de 2014].

La estructura fundamental del modelo relacional es la relación, es decir una tabla bidimensional constituida por filas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representan las propiedades de la entidad. Si esta estructura se puebla con datos, entonces tendremos una lista de valores individuales para cada tupla, atributo por atributo. Aunque una relación es más conocida como tabla, las tuplas como filas y los atributos como columnas.

Al igual que las bases de datos jerárquica y de red, la base de datos relacional es un depósito de datos en el que se mantiene su independencia; sin embargo el modelo relacional ofrece varias ventajas importantes:

- Diseño, ejecución y uso más fácil de la base de datos: dado que este modelo cuenta con independencia de los datos e independencia estructural, es más fácil el diseño y administración de su contenido.
- Un poderoso sistema de base de datos: El sistema de administración de base de datos relacional es una pieza de software mucho más sofisticada que el SGBD del modelo jerárquico y de red, dado que realiza tareas muchas más complejas disminuyendo la complejidad física del sistema tanto para el diseñador como para los usuarios finales.
- Capacidad de consulta: una de las principales razones de la predominante posición en el mercado del modelo relacional, es la capacidad de consulta con que se cuenta a través de un lenguaje de consulta estructurado SQL.
- SQL trabaja con consultas ad hoc a partir del lenguaje de cuarta generación (4GL) que le permite al usuario especificar qué debe hacer sin tener que decir cómo hacerlo. El sistema de administración de base de datos relacional utiliza SQL para transformar la consulta de usuario en el código tecnológico requerido para recuperar los datos solicitados, por lo tanto, la base de datos relacional requiere menos tiempo de desarrollo de código que otros modelos de bases de datos.

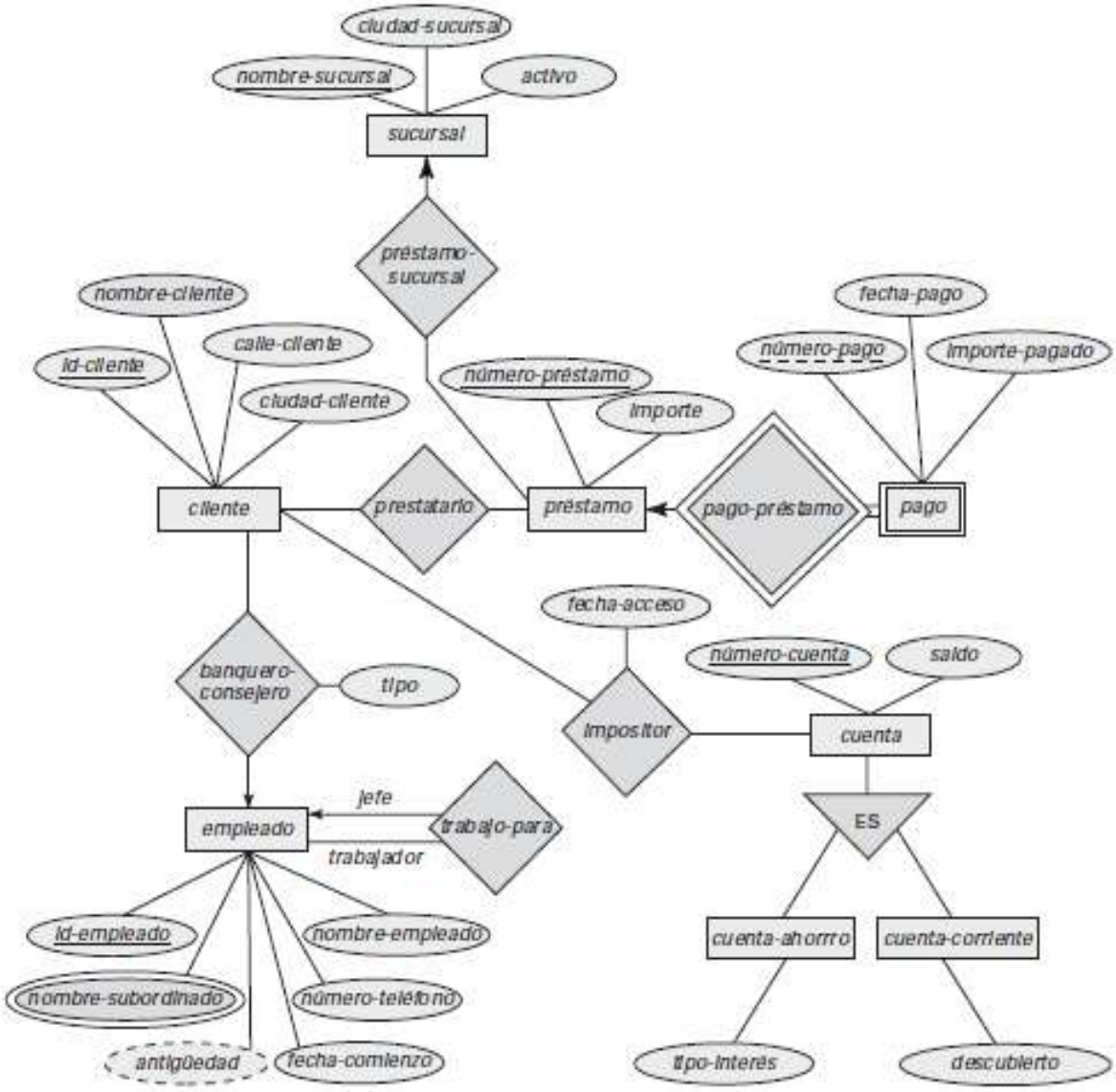
2.2.5. Modelo Entidad Relación. El modelo Entidad–Relación (ER) (también llamado entidad-interrelación) es el modelo conceptual más utilizado para el diseño conceptual de BD. Es una herramienta para el modelado de datos que describe mediante un conjunto de representaciones gráficas y lingüísticas la realidad. Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema que representa la estructura lógica completa de una base de datos. El modelo ER es extremadamente útil para representar los

significados e interacciones de las empresas del mundo real con un esquema conceptual.

El modelo ER está formado por un conjunto de conceptos (entidad, atributo, relación) que permiten describir la información relevante de la realidad mediante un conjunto de representaciones gráficas.

Fue introducido por Peter Chan en 1976 en el artículo The Entity–Relationship Model. Toward a Unified View of Data, Chen, P. Transactions on Database Systems, Vol.1, como un enfoque de modelado conceptual que trabaja los datos como sistemas de entidades y sus relaciones propias, llevándose a través del diagrama entidad relación a su forma gráfica.

Figura 7. Diagrama Entidad – Relación:



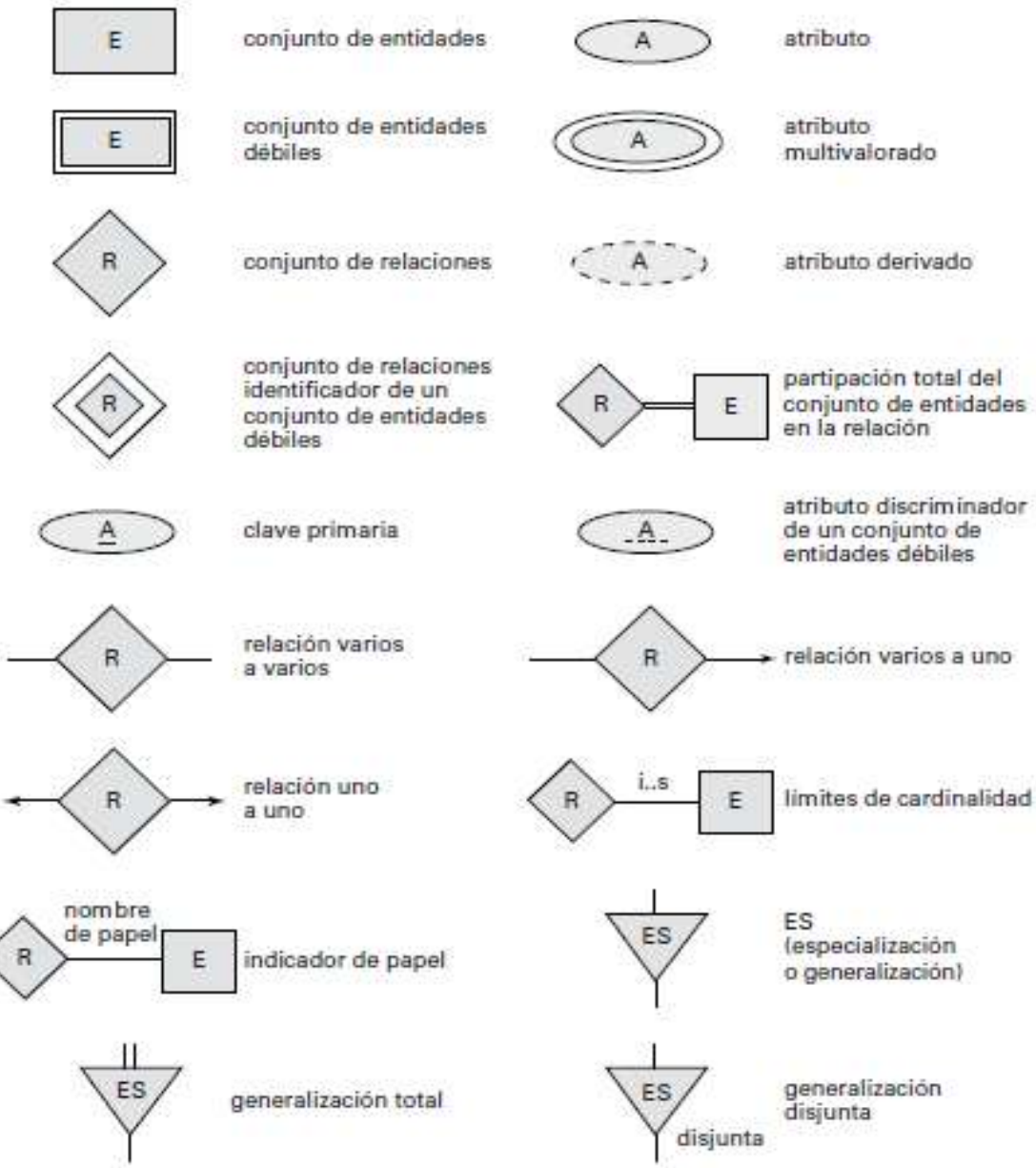
Fuente: SILBERSCHATZ, Abraham; KORTH, Henry F. y SUDARSHAN. Fundamentos de bases de datos. Cuarta edición. Madrid: McGraw-Hill, 2002. 119 p.

El diagrama entidad relación se define como un instrumento de gran valor para el modelado de relaciones entre las diferentes entidades, ya que le proporciona al

diseñador un bosquejo de la base de datos de una forma clara y precisa. Esta describe su estructura usando una serie de figuras y líneas, de forma didáctica y entendible para todos los profesionales en base de datos y otras áreas que tengan conocimientos básicos de entidades y relaciones, los diagramas son simples y claros, cualidades que pueden ser responsables del amplio uso del modelo Entidad - Relación. Este diagrama consta de los siguientes componentes principales:

- Rectángulos, que representan conjuntos de entidades.
- Elipses, que representan atributos.
- Rombos, que representan relaciones.
- Líneas, que unen atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.
- Elipses dobles, que representan atributos multivalorados.
- Elipses discontinuas, que denotan atributos derivados.
- Líneas dobles, que indican participación total de una entidad en un conjunto de relaciones.
- Rectángulos dobles, que representan conjuntos de entidades débiles.

Figura 8. Componentes del Diagrama Entidad-Relación



Fuente: SILBERSCHATZ, Abraham; KORTH, Henry F. y SUDARSHAN. Fundamentos de bases de datos. Cuarta edición. Madrid: McGraw-Hill, 2002. 119 p.

El modelo y diagrama entidad relación en conjunto, cuentan con las siguientes ventajas:

- Herramienta de comunicación efectiva: El modelo integra las diferentes visualizaciones de los datos convirtiéndose en una poderosa herramienta de comunicación, ya que permite que el diseñador de la base de datos capture los datos, tal como los programadores, gerentes y usuarios finales necesitan verlas.
- Representación visual: El modelo entidad-relación le proporciona a las personas que interactúan con la base de datos (programadores, diseñadores y usuarios finales), facilidad en la representación visual de los datos y sus relaciones.
- Simplicidad conceptual excelente: El modelo entidad – relación permite una fácil y agradable representación de las entidades principales y las relaciones de la base de datos lo que permite una excepcional visualización lógica de los datos que los demás sistemas de administración de archivos.

2.2.6. Normalización. Una vez obtenido el esquema relacional resultante del modelo entidad relación que representaba la base de datos, normalmente tendremos una buena base de datos. Pero otras veces, debido a fallos en el diseño o a problemas indetectables en esta fase del diseño, tendremos un esquema que puede producir una base de datos que incorpore estos problemas como redundancia, ambigüedades, pérdida de restricciones de integridad y anomalías en operaciones de modificación de datos.

El principio fundamental reside en que las tablas deben referirse a objetos o situaciones muy concretas. Lo que ocurre es que conceptualmente es difícil obtener ese problema. La solución suele ser las formas normales.

Las formas normales corresponden a una teoría de normalización iniciada por el propio Codd y continuada por otros autores (entre los que destacan Boyce y Fagin). La Normalización es un proceso que consiste en aplicar una serie de reglas que permitan organizar la base de datos, evitar la redundancia y por

consiguiente, garantizar la integridad de la información. Tener la base de datos relacionada y normalizada correctamente evitará redundancia de datos y proporciona una base de datos más ligera, más correcta y con menores tiempos de ejecución.

La normalización tiene como objetivo obtener esquemas relacionales que cumplan determinadas condiciones, a través de las siguientes formas normales:

2.2.6.1 Primera Forma Normal (1FN). Fue introducida por Codd, en su primer trabajo. Es una restricción inherente al modelo relacional por lo que su cumplimiento es obligatorio. Consiste en la prohibición de que en una relación existan grupos repetitivos, es decir, un atributo no puede tomar más de un valor del dominio subyacente, lo que en otras palabras se dice que para que una base de datos cumpla la primera forma normal, cada columna debe ser atómica.

Atómica significa "indivisible", es decir, que cada atributo debe contener un único valor del dominio. Los atributos no pueden tener listas o arrays de valores, ya sean del mismo dominio o de dominios diferentes. Además, cada atributo debe tener un nombre único.

2.2.6.2 Segunda Forma Normal (2FN). Fue introducida por Codd. Una relación está en 2FN, si además de estar en 1FN, la llave principal hace dependientes al resto de atributos, por lo tanto, si hay atributos que depende sólo de parte de la llave, entonces esa parte de la llave y esos atributos formarán otra tabla, por consiguiente, si la relación consta de un sólo atributo que corresponda a la llave, entonces automáticamente está en la segunda forma normal.

2.2.6.3 Tercera Forma Normal (3FN). Propuesta por Codd. Una relación está en 3FN, si además de estar en 2FN, todos los atributos que no forman parte de ninguna clave candidata facilitan información sólo acerca de la(s) clave(s) y no

acerca de otros atributos, lo que indica que es necesario eliminar cualquier relación que permita llegar a un mismo dato de dos o más formas diferentes.

2.2.6.4 Forma Normal de Boycce y Codd (FNBC). Una relación está en FNBC si y solo si, el conocimiento de las claves candidatas permite averiguar todas las interrelaciones existentes entre los datos de la relación, lo que significa que no deben existir interrelaciones entre atributos fuera de las claves candidatas.

2.2.6.5 Cuarta forma normal (4FN). Una relación está en la cuarta forma normal, si además de estar en FNBC, no tiene dependencias multivaluadas.

Una dependencia multivaluada se presenta cuando se toman tres atributos y existe dependencia funcional entre dos de ellos, mientras que el resto dependerá funcionalmente de uno de los dos y no de ambos.

2.2.6.6 Quinta forma normal (5FN). Fue definida por Fagin. La quinta forma hace referencia a las dependencias menos usuales en las bases de datos, lo que las convierte en las más complejas y polémicas. Estas tienen que ver con relaciones que pueden dividirse en subrelaciones, pero que no se pueden reconstruir y son utilizadas normalmente para eliminar dependencias de proyección o reunión y se deben a restricciones muy concretas.

3. DESARROLLO E IMPLEMENTACIÓN DE LA METODOLOGIA DEL PROYECTO

3.1. METODOLOGÍA SCRUM

Para el desarrollo de la herramienta software se planteó el estudio y utilización del Método Ágil Scrum.

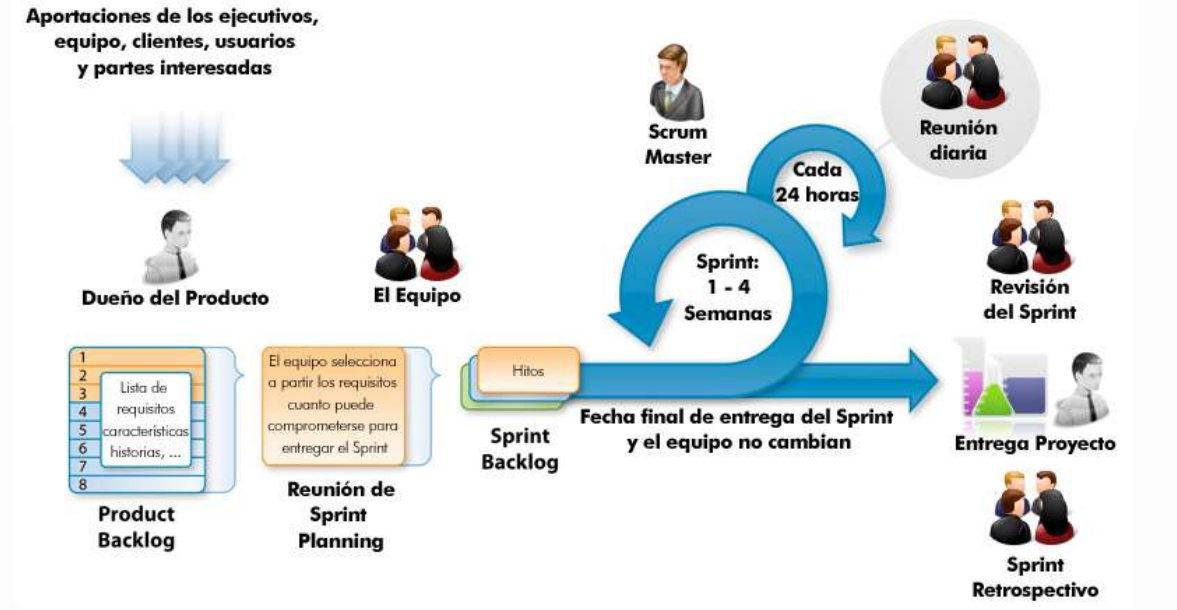
[5]Scrum es una metodología iterativa e incremental de gestión, mejora y mantenimiento de sistemas existentes o prototipos de producción. Fue implementado como respuesta contra las prácticas de programación demasiado estructuradas y estrictas, que convertían el proceso de desarrollo en algo burocrático, lento, degradante e inconsistente con las expectativas de eficiencia de desarrolladores y clientes.

El objetivo principal de Scrum es incrementar al máximo nivel la productividad del equipo de desarrollo, reduciendo las actividades no orientadas a producir software funcional y dando resultado en periodos de tiempo más corto de lo usual.

Su forma de trabajo esta enfatizada en valores y prácticas de gestión, sin tratar sobre prácticas de desarrollo, implementación y demás cuestiones técnicas, por lo que delega al equipo de trabajo la responsabilidad de trabajar de la mejor manera y lo más productivo posible.

[5] ANGARITA, Heiner y TOSCANO, Luis Gabriel. Prototipo de sistema de información web para la gestión de proyectos software bajo la metodología Scrum. Universidad Industrial de Santander. 2013. 85 p.

Figura 9. Método Ágil Scrum



Fuente: Desarrollo y diseño Web. Disponible en internet: http://www.islavisual.com/articulos/desarrollo_web/diferencias-entre-scrum-y-xp.php [Consultado 02 de Febrero de 2014].

La ejecución del proyecto permitió evidenciar las fortalezas y debilidades de la utilización de Scrum en proyectos de desarrollo de software, además constituyo un aporte significativo en el estudio y utilización de metodologías agiles de desarrollo como alternativa a las metodologías tradicionales.

A continuación será analizada la aplicación de la metodología Scrum a la herramienta software desarrollada en este proyecto:

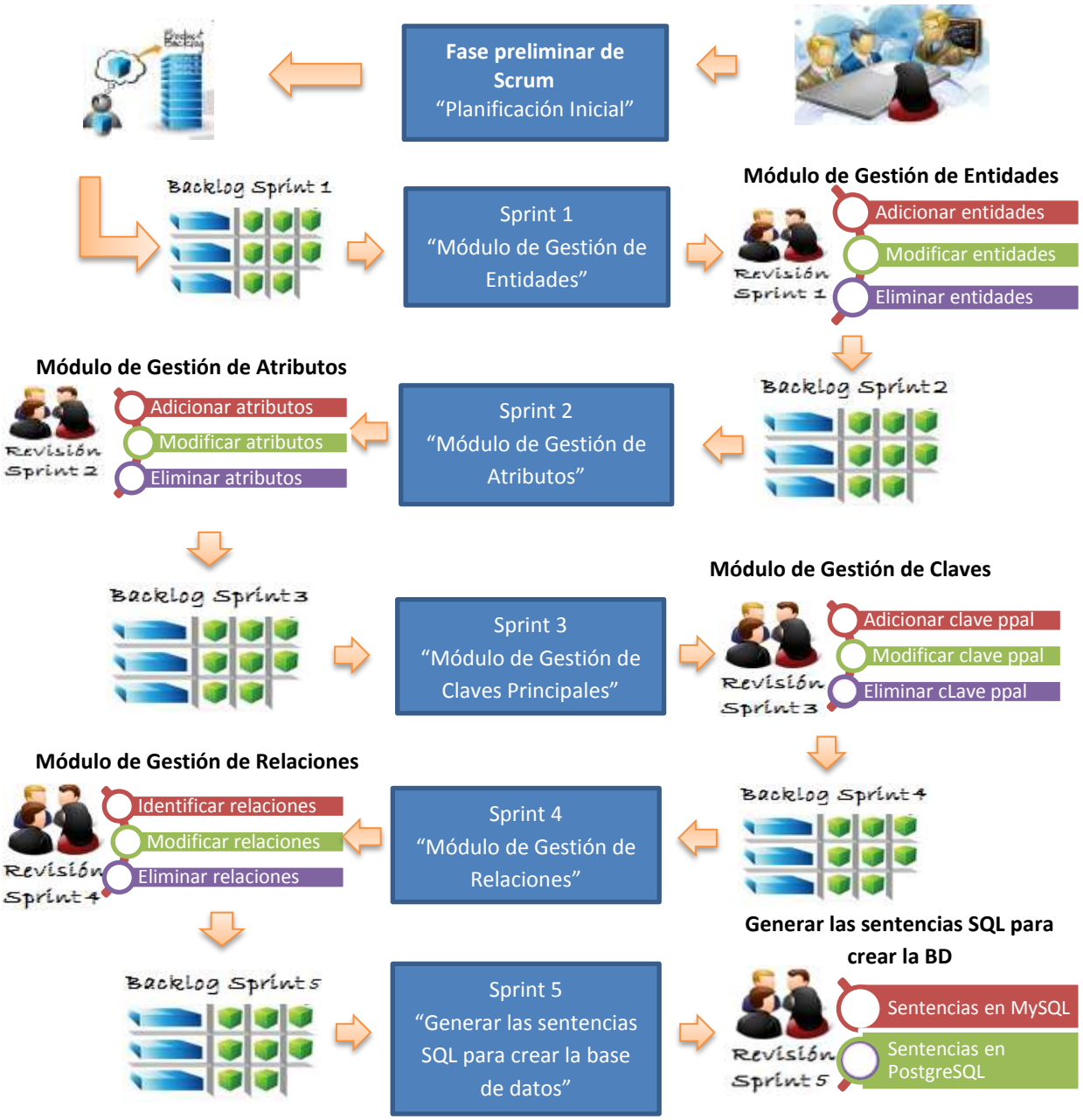
3.1.1. Desarrollo iterativo e incremental. Scrum basa su metodología en un desarrollo incremental e iterativo, partiendo de la posibilidad de variaciones que se puedan presentar durante el proyecto.

En la fase preliminar se desarrollan cada uno de los eventos, siguiendo la metodología (reunión con el cliente, creación del Listado del Producto, estimación de niveles de esfuerzo para elementos en el listado, estimación de duración del proyecto, establecimiento de duración de iteraciones y planeación de la primera iteración del proyecto) todas las anteriores realizadas antes de la primera iteración.

Durante la fase preliminar, se planificaron cinco iteraciones (Sprints), cada una con una duración de dos semanas. Cada Sprint busca incrementar funcionalidades agrupadas en módulos de la aplicación. La planificación inicial permitió definir el Backlog del producto, el cual se constituyó en base a los Backlogs de cada Sprint.

La finalización de cada Sprint arrojo como resultado una versión estable del producto, con el incremento de las funcionalidades planificadas, las mismas que fueron presentadas por el Product Ower. Para ello, se trabajó de tal forma que cada requisito planificado se completará en una única iteración (incluyendo pruebas y documentación).

Figura 10. Metodología Scrum aplicada al desarrollo del proyecto



3.1.2. Fases. Cada sprint o iteración que conformará el proyecto puede tomarse como un proyecto individual, en cada sprint se realiza una nueva subdivisión del trabajo con el fin de garantizar un resultado final más completo y de esta forma el Product Owner obtenga los beneficios de forma incremental.

Cada sprint se desarrolló en cinco fases muy similares al modelo en cascada de la siguiente forma:

- **Modelado del negocio:** Esta fase se encarga de comprender y describir la realidad del negocio. Se realiza primordialmente durante la fase de inicial, debido a que en ella se analiza el negocio, los requerimientos y se planifica el trabajo a realizar en los Sprints 1,2, 3, 4 y 5.
- **Requisitos:** Culminada satisfactoriamente la etapa preliminar, se da paso a la definición de requisitos para el desarrollo del sistema. Esta fase tiene como principal objetivo especificar las funcionalidades de la herramienta que serán implementadas durante cada Sprint como resultado de las reuniones realizadas con el Product Ower.
- **Análisis y diseño:** A partir de los resultados anteriores, se busca hacer un análisis del resultado final esperado pero a nivel funcional y los requisitos levantados en la fase anterior, priorizarlos y llevarlos al lenguaje del programador y trabajar en un diseño que represente las características requeridas a una implementación más adecuada.
- **Implementación:** En esta fase, el equipo de desarrollo implementa las funcionalidades necesarias, de acuerdo a los requerimientos propuestos y analizados y según el diseño anteriormente planteado.
- **Pruebas:** Esta etapa consiste en realizar las pruebas que garanticen el correcto funcionamiento de las funcionalidades implementadas. En esta etapa interactúa tanto el equipo de desarrollo como el cliente del producto, quien realiza una revisión de la herramienta y su desenvolvimiento frente a los requisitos por él solicitados.

3.1.3. Diseño e Implementación. Como resultado de la fase preliminar de Scrum en donde se analizaron los requerimientos dados por el dueño del producto, se planificó el desarrollo de la herramienta en cinco sprints, los cuatro primeros con el objetivo de implementar los cuatro módulos que conforman el paso a paso de la

metodología usada en la herramienta y el ultimo con la generación de las sentencias SQL necesarias para la creación de la base de datos.

- Sprint 1: Módulo de Gestión de Entidades
- Sprint 2: Módulo de Gestión de Atributos
- Sprint 3: Módulo de Gestión de Claves principales
- Sprint 4: Módulo de Gestión de Relaciones
- Sprint 5: Generación de las sentencias SQL para crear la Base de Datos

Las funcionalidades que se van a implementar en cada Sprint se describieron a partir de las historias de usuario priorizadas en la fase inicial según la importancia dada por el Product Owner y analizadas según las especificaciones el producto. De ahí se desarrolló la arquitectura predominante para la construcción del software a partir de 4 módulos que están ligados directamente a los cuatro primeros sprints y el último sprint que será el que arroje finalmente el resultado esperado de la herramienta.

3.1.3.1. Descripción de los Módulos del Sistema

Figura 11. Módulos del Sistema

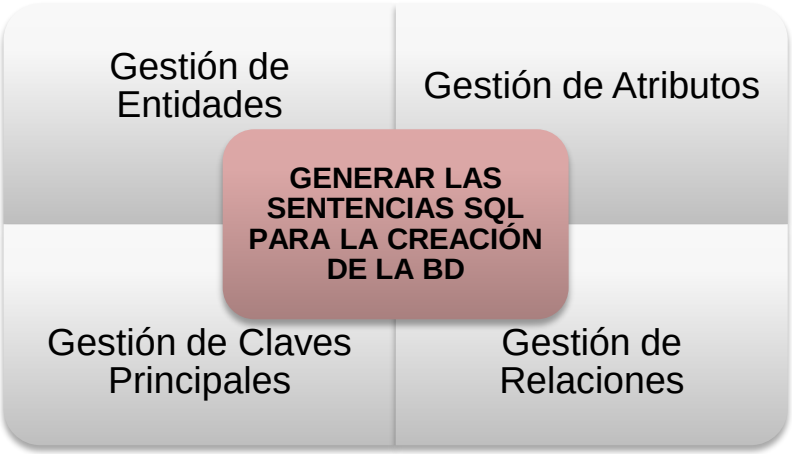


Tabla 1. Descripción de los Módulos del Sistema

Módulo	Descripción
Gestión de Entidades	Este módulo agrupa las funcionalidades que permiten la administración de las Entidades, incluye la adición, modificación o eliminación de entidades, permitiéndole al usuario interactuar con ellas.
Gestión de Atributos	Este módulo agrupa las funcionalidades de administración de los Atributos, permite la adición, modificación o eliminación de atributos de cada una de las entidades añadidas en el módulo anterior.
Gestión de Claves Principales	Este módulo agrupa las funcionalidades de administración de las Claves Principales, permite la selección, modificación o eliminación de claves principales partiendo de los atributos ingresados en el módulo anterior.
Gestión de Relaciones	Este módulo agrupa las funcionalidades de administración de las Relaciones, permite la adición, modificación o eliminación de relaciones, además de la asignación de la cardinalidad de estas. Si existe una relación mucho a muchos se procede a la creación de una nueva tabla en la que será necesario definir tanto el nombre de esa nueva entidad, los atributos y la(s) claves principales.
Generación de las Sentencias SQL para la creación de la Base de Datos	Este es el resultado de la herramienta, aquí se generan las sentencias de creación de la Base de Datos, ya sea en MySQL o en PostgreSQL, dependiendo de la selección hecha por el usuario durante del uso de la herramienta.

3.1.3.1.1. Módulo de Gestión de Entidades

Funcionalidades	Descripción
Adicionar Entidades	Esta funcionalidad le permite al usuario crear la cantidad de entidades que desee y asignarle el nombre a cada una.
Modificar Entidades	Esta funcionalidad permite renombrar las entidades anteriormente creadas.
Eliminar Entidades	Esta funcionalidad le permite al usuario eliminar una(s) de las entidades creadas anteriormente. Es necesario aclarar que la base de datos debe contener como mínimo dos entidades para que sea posible crear una base de datos relacional.

3.1.3.1.2. Módulo de Gestión de Atributos

Funcionalidades	Descripción
Adicionar Atributos	Esta funcionalidad permite que el usuario cree la cantidad de atributos que desee a cada entidad, además permite asignarle el nombre y el tipo de dato a cada uno.
Modificar Atributos	Esta funcionalidad permite renombrar los atributos anteriormente creados, además de la modificación del tipo de dato, si es necesario.
Eliminar Atributos	Esta funcionalidad permite que el usuario elimine uno(s) de los atributos creados anteriormente, es necesario aclarar que cada entidad debe tener como mínimo un atributo.

3.1.3.1.3. Módulo de Gestión de Claves Principales

Funcionalidades	Descripción
Adicionar Claves Principales	Esta funcionalidad permite que el usuario seleccione la(s) claves principales de cada entidad partiendo de los atributos definidos en el módulo anterior.
Modificar Claves Principales	Esta funcionalidad permite modificar la selección de la(s) claves principales realizada anteriormente, además permite seleccionar un atributo nuevo como clave principal.
Eliminar Claves Principales	Esta funcionalidad le permite al usuario eliminar la selectividad de las claves principales realizadas anteriormente, es necesario aclarar que cada entidad debe tener como mínimo un atributo que corresponda a clave principal.

3.1.3.1.4. Módulo de Gestión de Relaciones

Funcionalidades	Descripción
Adicionar Relaciones	Esta funcionalidad permite que el usuario seleccione la(s) entidades que tienen relación directa, así mismo, permite definir el grado de cardinalidad de la relación.
Modificar Relaciones	Esta funcionalidad permite modificar las entidades que tienen relación directa y redefinir el grado de cardinalidad de las relaciones definidas anteriormente
Eliminar Relaciones	Esta funcionalidad le permite al usuario eliminar las relaciones definidas anteriormente. Es necesario aclarar que para que exista una base de datos relacional debe existir mínimo una relación entre dos entidades.

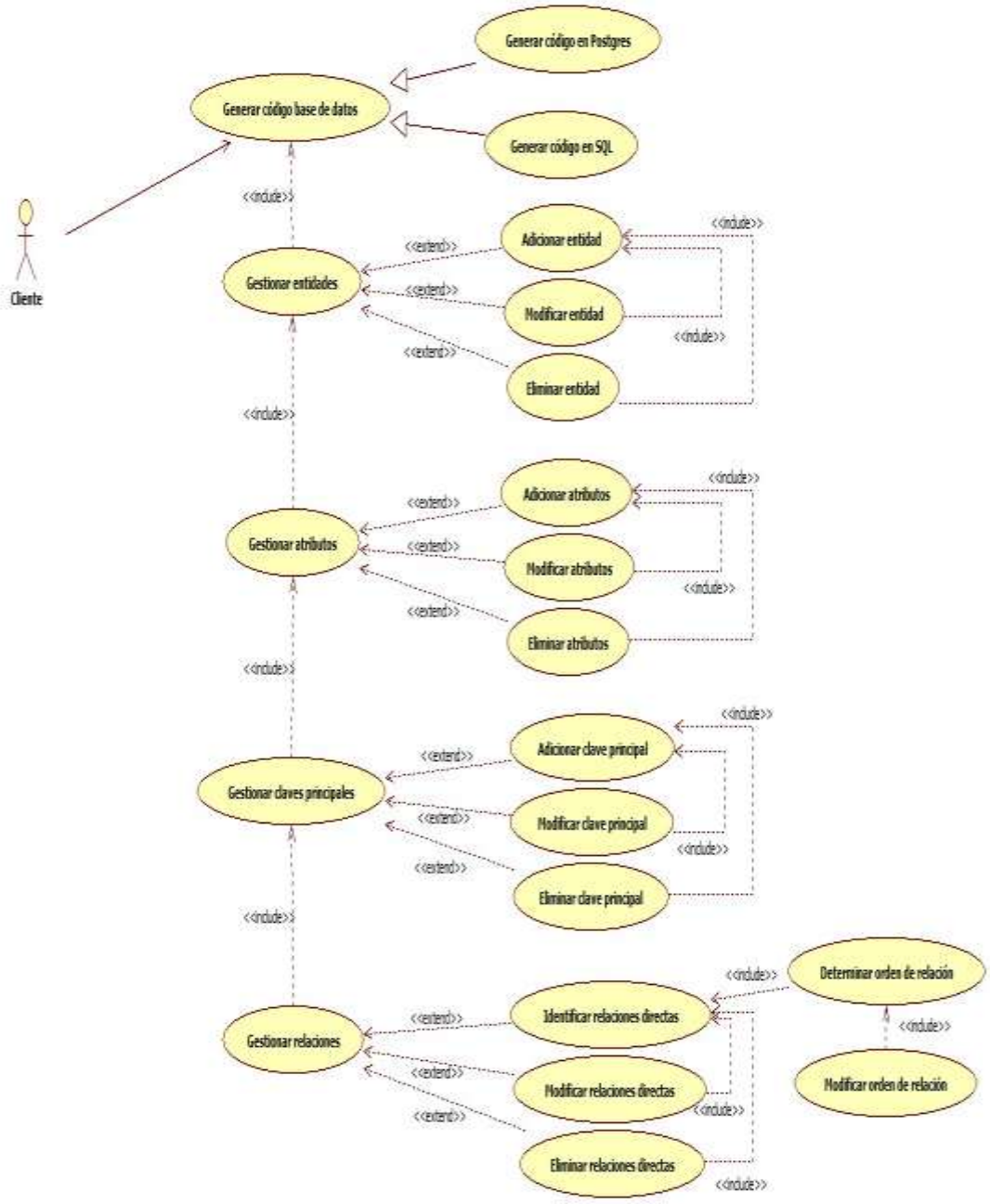
3.1.3.1.5. Generación de las Sentencias SQL para la creación de la Base de Datos

Funcionalidades	Descripción
Generación de Sentencias en MySQL para la creación de la Base de Datos	Esta funcionalidad permite generar las sentencias para crear la base de datos en MySQL, partiendo de la elección hecha por el usuario en el primer pantallazo de la herramienta.
Generación de Sentencias en PostgreSQL para la creación de la Base de Datos	Esta funcionalidad permite generar las sentencias para crear la base de datos en PostgreSQL, partiendo de la elección hecha por el usuario en el primer pantallazo de la herramienta.

3.1.4. Entregables del proyecto. A continuación se indican y describen cada uno de los entregables generados durante la ejecución del proyecto:

3.1.4.1. Diagrama de Casos de Uso del Negocio. A través del diagrama de casos de uso, se modelaron todas las funciones de la herramienta vistas desde la perspectiva del usuario final, quien será el único actor que finalmente interactuará con la herramienta.

Figura 12. Diagrama de Casos de Uso



3.1.4.2 Diseño de las Interfaces de Usuario. Se diseñaron alrededor de 8 interfaces de usuario que permitan visualizar la interacción entre los diferentes

componentes del sistema y el usuario final, ahí se representa gráficamente los 4 módulos que formarán parte de la herramienta además de otras características que fueron solicitadas por el cliente del producto. El diseño de las interfaces se realizó teniendo como prioridad los requerimientos planteados para la herramienta, estas características se pueden organizar así:

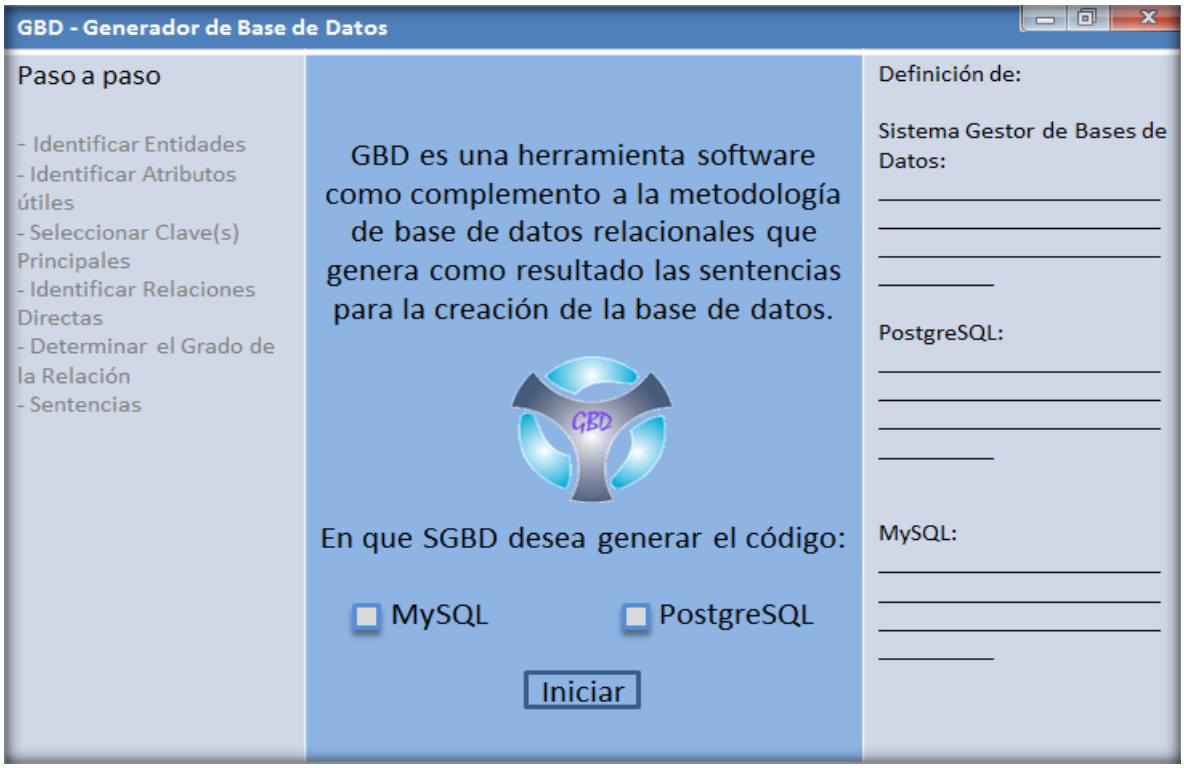
- Claridad: Es necesario que cada una de las interfaces que conformarán la herramienta sean lo más claras posible, para ellos es necesario que sea visualmente comprensible y su funcionalidad sea evidente.

- Fácil usabilidad: Es necesario que el manejo y control que el usuario tenga sobre la herramienta permita el máximo aprovechamiento de esta, para ellos es necesario que la distribución de los mensajes descriptivos e informativos, las imágenes, botones, texto, y demás componentes de las interfaces sean consistentes y faciliten el total entendimiento del proceso a desarrollar.

- Estética: Las interfaces que conforman la herramienta procuran ser estéticamente llamativas, para ello se usará una gama de colores, simetría, sentido de proporción, saturación de información y relevancia que sea visualmente agradable y cautivadora para los usuarios.

- Consistencia: El flujo de pantallas y su distribución durante el uso de la herramienta debe ser consecuente y entendible para el usuario partiendo de la realidad; los títulos y texto que guíen el proceso desarrollado en la herramienta deben simples y comprensibles para todo tipo de usuarios.

Figura 13. Interfaz uno y dos del Sistema



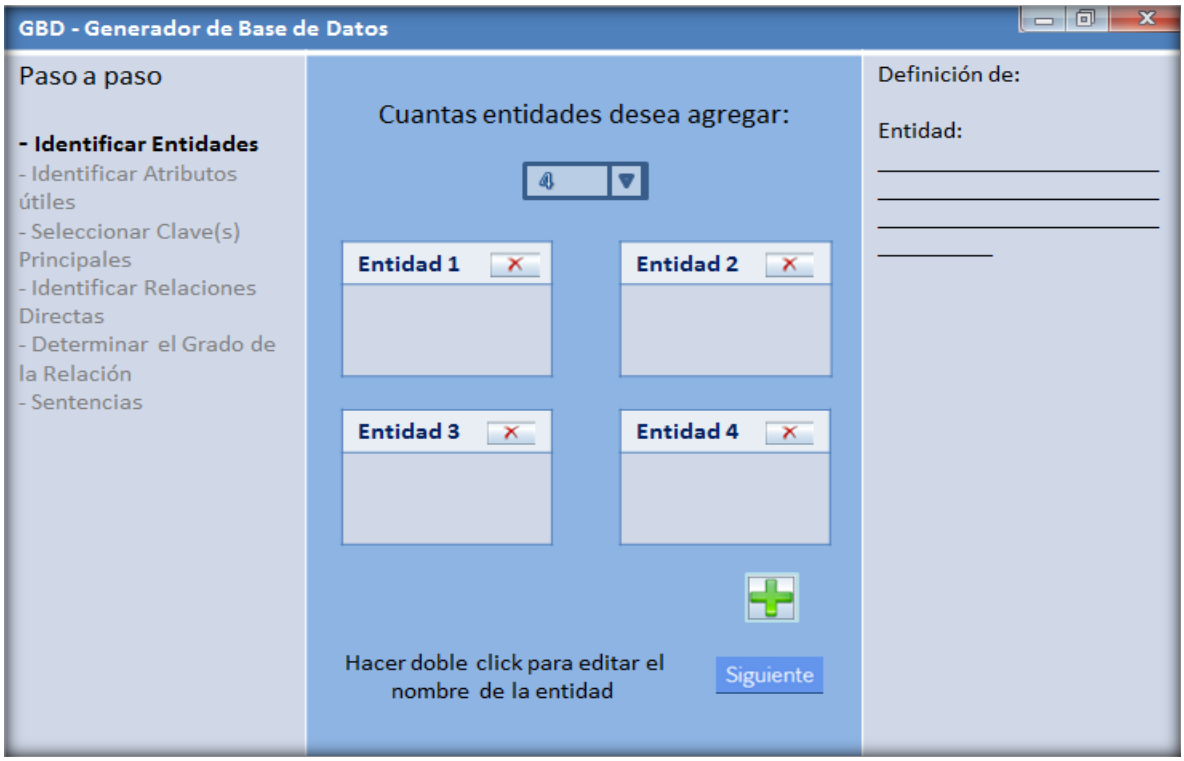
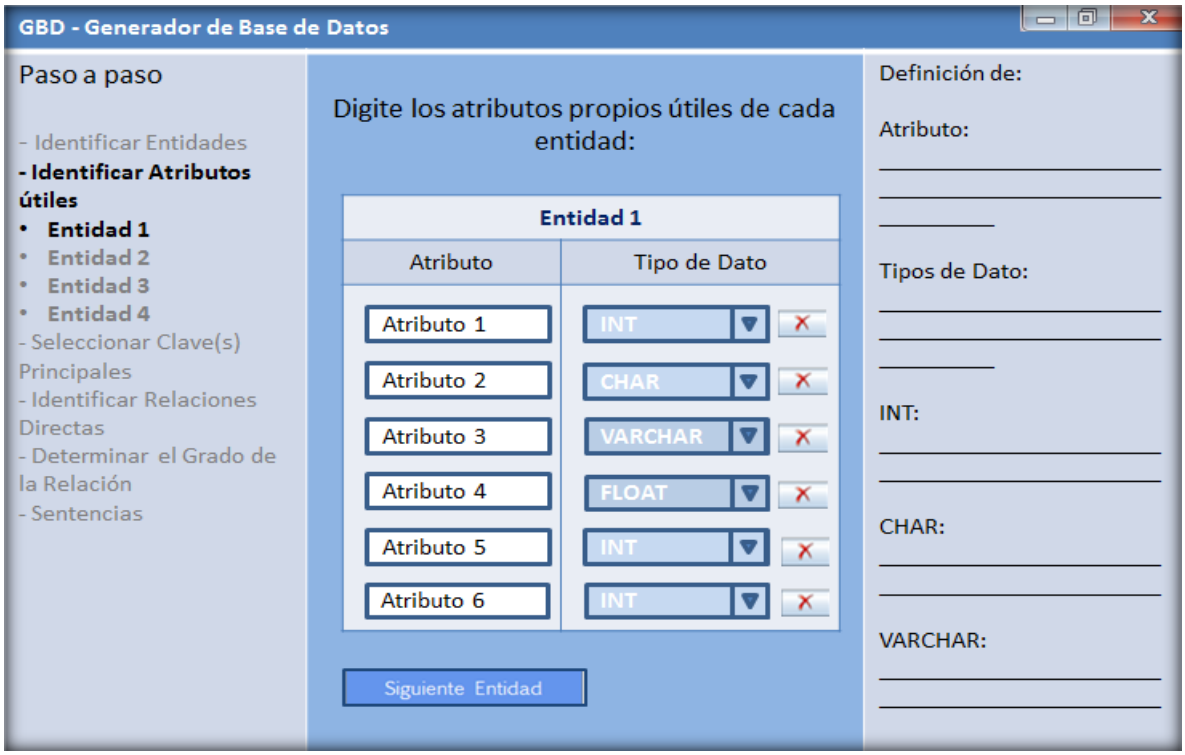


Figura 14. Interfaz tres y cuatro del Sistema



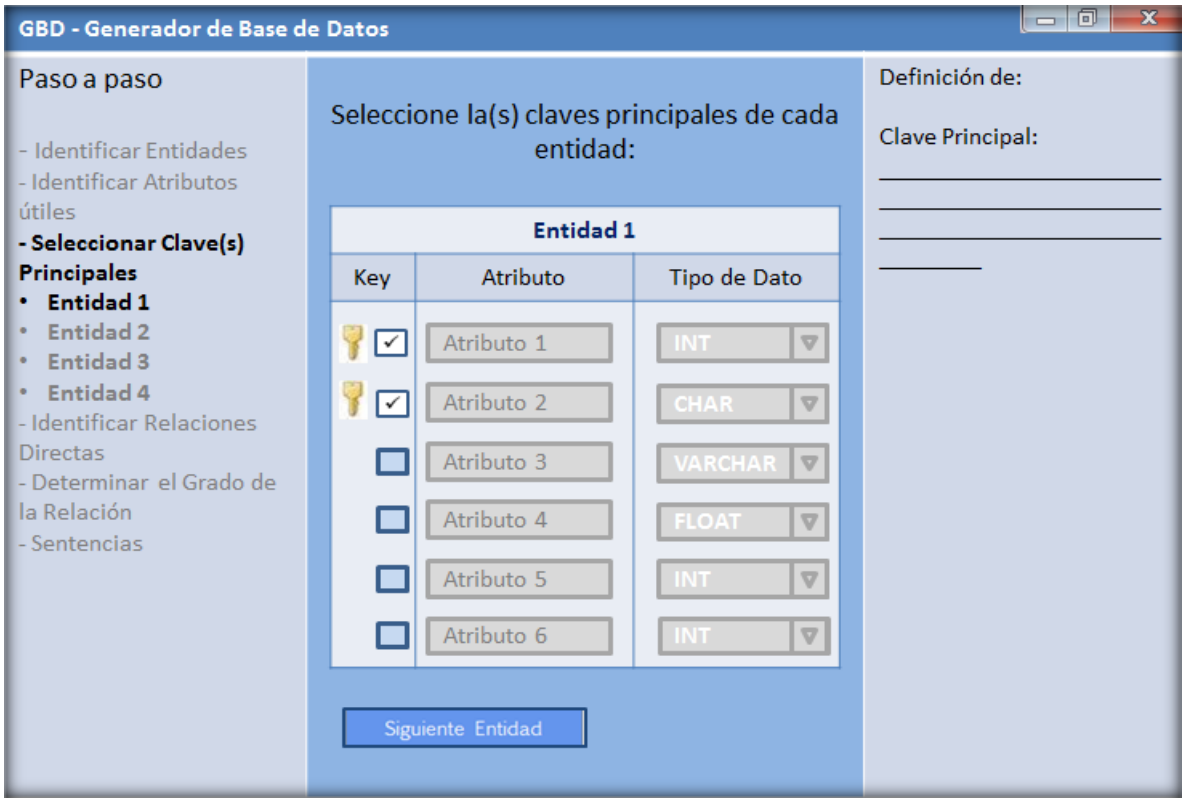
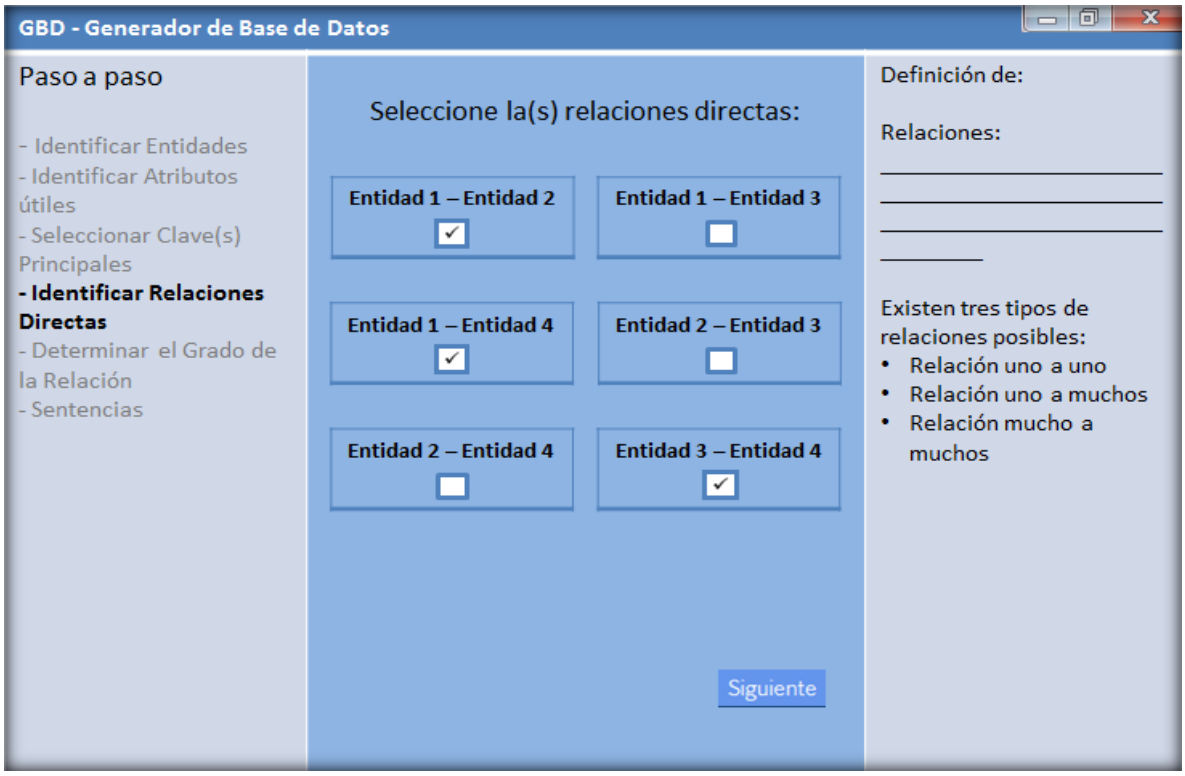


Figura 15. Interfaz cinco y seis del Sistema



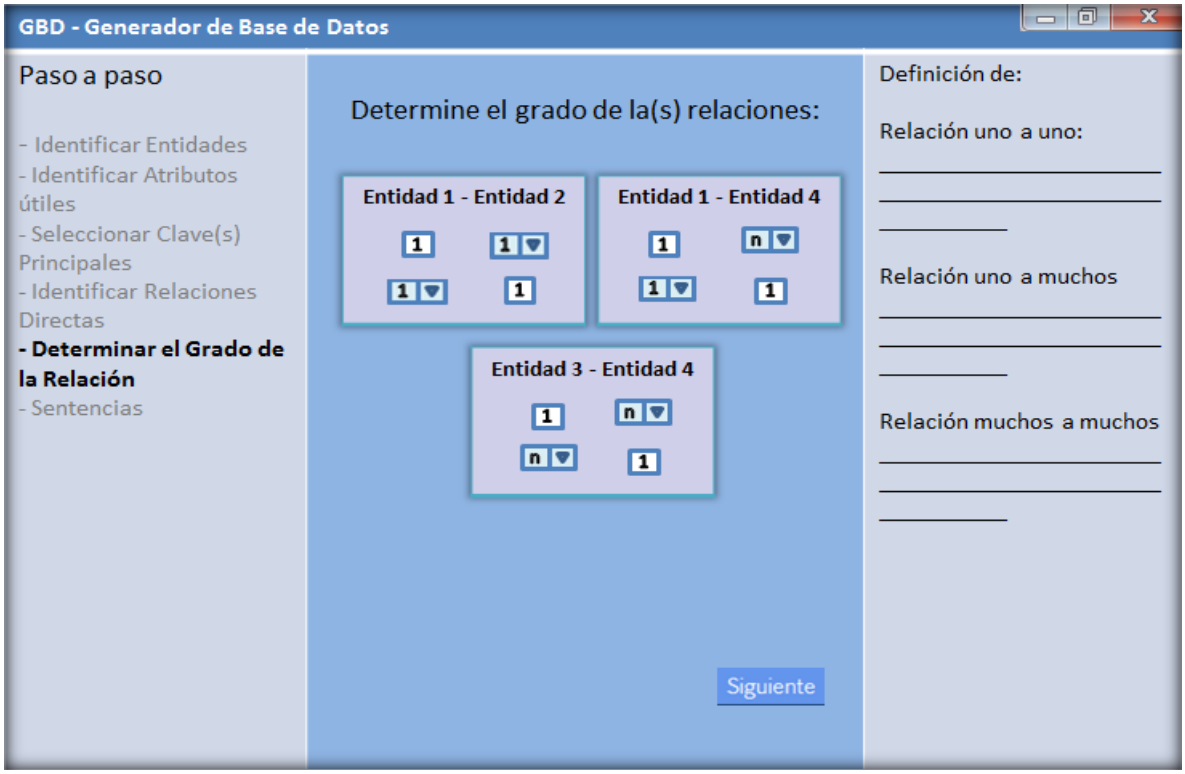
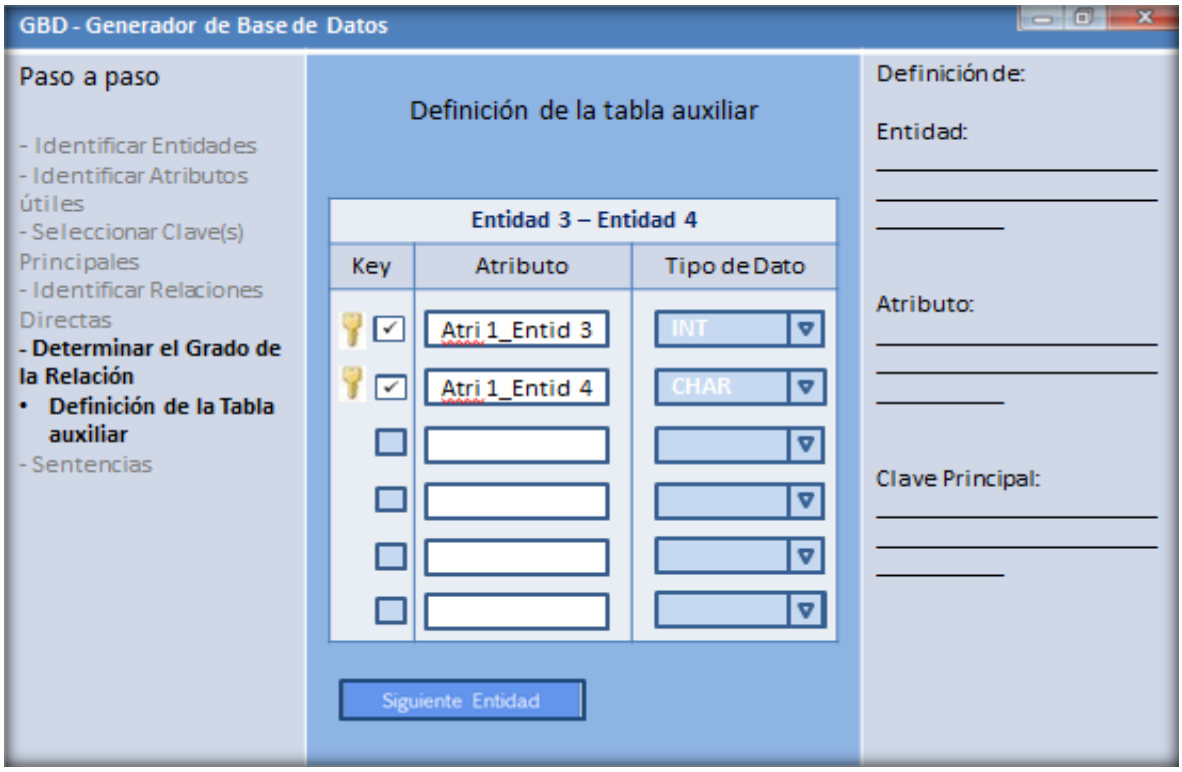
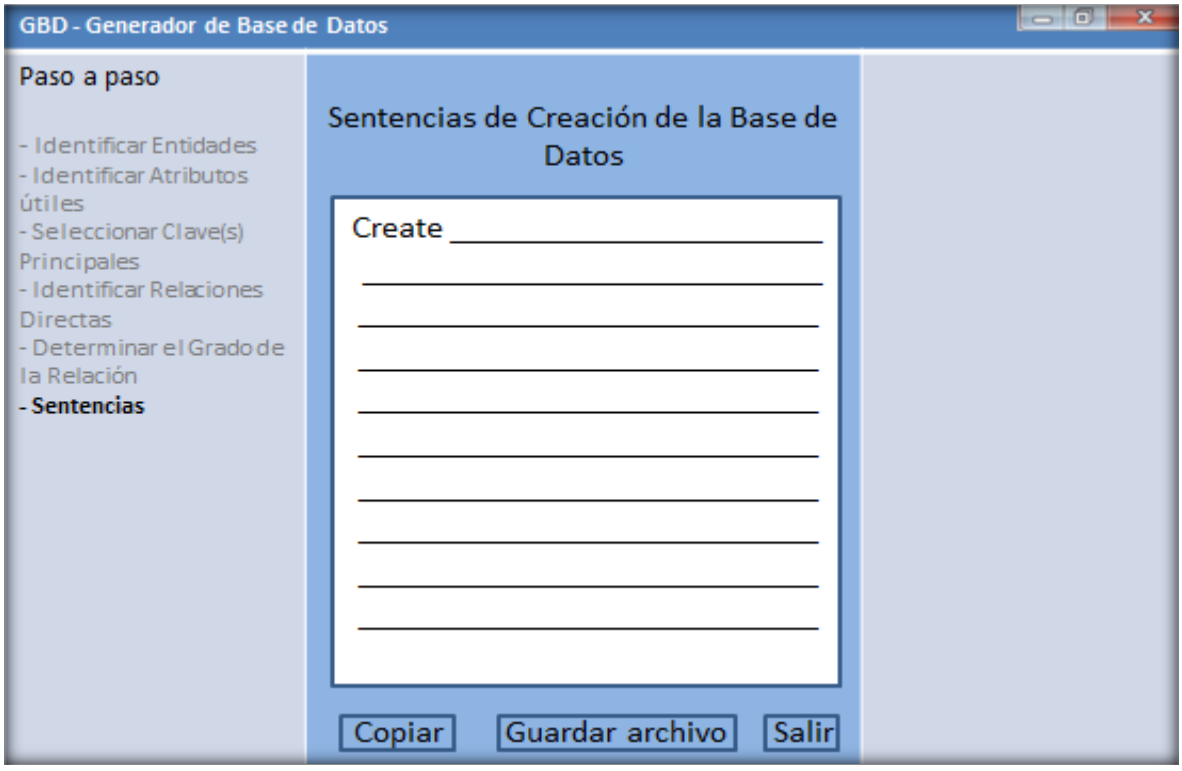


Figura 16. Interfaz siete y ocho del Sistema





3.1.4.3 Product Backlog (Listado del producto). El product backlog consiste principalmente en las historias de usuario convertidas en elementos del Listado del Producto para que el equipo de trabajo se encargue de implementarlas. A continuación se describe el listado del producto, a cada tarea se le asigna la prioridad según la importancia identificada y se estima el tiempo que se requerirá para implementar la tarea:

Tabla 2. Listado del Producto

Descripción de requisitos, prioridades y tiempos estimados			
Requisito	**Descripción**	**Prioridad de Desarrollo**	**Tiempo estimado (días)**
Abrir proyecto	El sistema permitirá abrir un proyecto existente.	MEDIA	2
Nuevo proyecto	El sistema permitirá crear un nuevo proyecto.	ALTA	2
Adicionar entidades	El sistema permitirá adicionar la cantidad de entidades que el usuario desee y asignarle el nombre a su preferencia.	ALTA	3
Modificar entidades	El sistema permitirá modificarle el nombre a las entidades ya creadas anteriormente.	BAJA	2
Eliminar entidades	El sistema permitirá eliminar las entidades que el usuario desee en cualquier momento.	MEDIA	2
Adicionar atributos	El sistema permitirá adicionar la cantidad de atributos a cada entidad que el usuario considere necesarios y asignarle el nombre de su preferencia.	ALTA	3
Modificar atributos	El sistema permitirá modificarle el nombre o el tipo de dato a los atributos propios de cada entidad definidos anteriormente.	BAJA	2
Eliminar entidades	El sistema permitirá eliminar los atributos de cada entidad, justo en el momento que usuario desee.	MEDIA	2
Adicionar claves principales	El sistema permitirá seleccionar la(s) claves primarias que el usuario	ALTA	3

Descripción de requisitos, prioridades y tiempos estimados			
Requisito	**Descripción**	**Prioridad de Desarrollo**	**Tiempo estimado (días)**
	considere necesarias a cada entidad.		
Modificar claves principales	El sistema permitirá modificar la(s) clave principal seleccionadas anteriormente y realizar una nueva elección de clave principal de acuerdo a las necesidades del usuario.	BAJA	2
Eliminar claves principales	El sistema permitirá eliminar la(s) claves principales ya seleccionadas, teniendo como única restricción, que cada entidad debe tener como mínimo una clave principal.	MEDIA	2
Adicionar relaciones	El sistema permitirá seleccionar que dupla de entidades están relacionadas directamente y así mismo seleccionar la cardinalidad de la relación.	ALTA	3
Modificar relaciones	El sistema permitirá modificar que dupla de entidades están relacionadas directamente y al mismo tiempo, modificar la cardinalidad de la relación.	BAJA	2
Eliminar relaciones	El sistema permitirá eliminar las relaciones ya definidas anteriormente, en el instante en que el usuario desee.	MEDIA	2
Generar las sentencias SQL para crear la base de datos	El sistema permitirá como resultado de los pasos anteriormente definidos, generar las sentencias SQL necesarias para la creación de la base de datos, partiendo del SGBD que el usuario desee.	ALTA	4
Guardar proyecto	El sistema permitirá guardar en cualquier momento, el trabajo realizado en la herramienta.	MEDIA	4

3.1.5. Producto. Los archivos del producto fueron empaquetados y archivados según las normas establecidas por la Biblioteca para entrega de proyectos de

grado, ahí estará contenida toda la herramienta software que se desarrolló a lo largo del proyecto.

3.1.6. Manuales. Sobre la versión final del producto, se generó el respectivo manual que permita al usuario la fácil utilización de la herramienta software implementada, además de la teoría necesaria para que el usuario logre obtener el mayor provecho de la herramienta.

4. EL PRODUCTO Y SU DOCUMENTACIÓN

4.1 DESARROLLO DE LA METODOLOGÍA DE BASES DE DATOS RELACIONALES APLICADA EN LA HERRAMIENTA SOFTWARE

En la actualidad, las empresas cuentan con un arma de gran poder para la organización, los datos. Estos datos bien gestionados y administrados pueden lograr un sin número de beneficios para las mismas.

Hoy en día todas las empresas cuentan con herramientas informáticas de creación de bases de datos, pero ¿por qué se producen fallos? La respuesta no está en las herramientas en sí, sino en cómo se diseña la base de datos. Cada herramienta dispone de sus propios utensilios de diseño, pero todos ellos se basan en los mismos conceptos teóricos, conceptos que si se desconocen no pueden ser aplicados correctamente y su utilización puede fracasar.

Si bien, existen distintas metodologías para el diseño de modelos de bases de datos y todos están orientados a obtener un modelo de datos funcional y consistente con la realidad, todos los métodos no son igualmente aplicables y prácticos, radicando su diferencia en la complejidad y comprensión de cada uno.

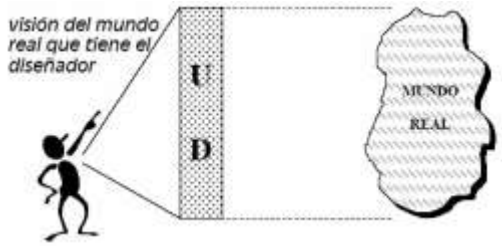
El método propuesto para el diseño del modelo de base de datos relacionales aplicado por la herramienta software, consiste en un conjunto de conceptos y reglas que permiten formalizar la semántica del mundo real que se pretende modelar a partir de una serie de pasos didácticos, que resultan bastante sencillos, prácticos de aplicar y compresibles para todo tipo de usuarios y que generan

finalmente, las sentencias de creación de la base de datos. Las actividades a desarrollar son:

4.1.1. Identificar el Universo. Todas las bases de datos deben tener un alcance (limitado) funcional: están dirigidas a una farmacia, a una universidad, a una petrolera, etc. Esta parte de la realidad que almacenamos en la base de datos se llama universo.

El "universo" puede definirse como una descripción abstracta y general de la parte o sector del universo real que el contenido de la base de datos va a representar. Por ello el diseñador de la base de datos debe establecer el contorno del problema; es decir “lo que forma y lo que no forma parte del problema”, por lo que es de gran importancia definir claramente este contorno.

Figura 17. Visión modo real que tiene el diseñador



Fuente: Bases de Datos. Modelos de Datos. Disponible en internet: http://www.lsi.us.es/docencia/get.php?id=1456 [Consultado 20 de Febrero de 2014].

4.1.2. Identificar entidades útiles. En primer lugar hay que definir los principales objetos que formarán parte de la base de datos, estos objetos se conocen como entidades.

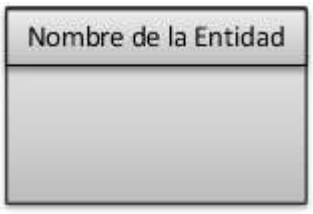
El método recomienda participar ya sea activamente en los procesos o como observador directo, al observar es posible notar las entidades que formaran parte de la base de datos.

Se define como Entidad a cualquier objeto, real o abstracto, que existe en un contexto determinado o puede llegar a existir y del cual deseamos guardar información.[6] Una entidad es una cosa u objeto en el mundo real que es distinguible de todos los demás objetos y que tiene un conjunto de propiedades que pueden identificar una entidad de forma unívoca.

Una entidad caracteriza a un tipo de objeto, real o abstracto, del problema a modelar, además tiene existencia propia, lo que significa que es distinguible del resto de las entidades, tiene nombre y posee atributos definidos en un dominio determinado. Una entidad puede ser concreta, como una persona o un libro, o puede ser abstracta, como un préstamo, unas vacaciones o un concepto, que es relevante para un sistema dado.

Las entidades se suelen representar usando tablas de la siguiente forma:

Figura 18. Representación de una entidad



4.1.3. Identificar atributos útiles y asociarlos a cada entidad. Habiendo identificado las entidades útiles, el siguiente paso consiste en identificar los atributos y asociarlos a cada entidad. Aquellas características de la entidad para

[6] AMAYA, Jairo. Sistemas de información gerenciales. Segunda edición. Bogota: ECOE Ediciones, 2009. 228 p.

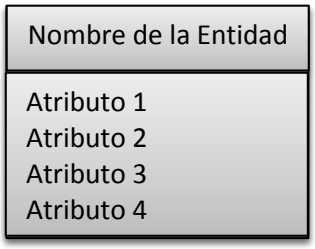
las cuales es importante registrar su valor o cambio en ellas; son los atributos útiles.

Se define como atributo a las propiedades, cualidades, identificadores o características de las entidades. Por ello, para identificar los atributos propios de cada entidad, es necesario preguntarse qué información se quiere saber de ellas y de esta forma reconocer si son simples o compuestos.

Los atributos simples están conformados por un solo elemento, los atributos grupales son combinaciones de otros atributos. Conforme se van identificando los atributos, se les asignan nombres que tengan significado para el usuario. De cada atributo se debe tener presente la siguiente información:

- Nombre y descripción del atributo.
- Tipo de dato y longitud.
- Valores por defecto del atributo (si se especifican).

Las entidades se representan en rectángulos verticales, el nombre de la entidad aparece en la parte superior y los atributos están descritos en orden, después del nombre de la entidad.



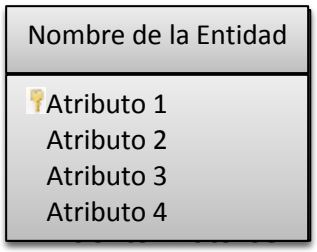
4.1.4. Seleccionar clave principal. En este paso se busca encontrar la(s) claves principales de cada una de las entidades, esto se hará escogiendo uno(s) de los atributos que forman parte de la entidad, como clave(s) primaria(s). Las claves principales o primarias permiten identificar en forma univoca a cada entidad.

Una clave principal es un atributo utilizado para clasificar y / o identificar los datos de la entidad de alguna manera. Como su nombre lo indica, son una parte clave de una base de datos relacional y una parte vital de la estructura la base de datos. Las llaves ayudan a hacer cumplir la integridad y se utiliza en la base de datos para ayudar a establecer relaciones con otras tablas**.** Como su nombre lo indica la clave principal debe contener valores únicos, nunca debe ser nula y debe identificar de forma exclusiva cada registro de la tabla. Una clave de una entidad debe cumplir con la siguiente condición:

"No pueden existir dos ocurrencias de la entidad con el mismo valor de la clave principal"

Las claves principales se representan con una llave ubicada frente al atributo denominado clave principal.

Figura 19. Representar Clave principal



4.1.5. Identificar las relaciones directas. Una vez definidas las entidades y sus atributos, se debe definir las relaciones que existen entre ellas, dándole importancia a las relaciones que son necesarias.

En este paso, se arman todas las posibles combinaciones en pares de las entidades útiles encontradas y por cada pareja, se analiza si la relación es directa o no, teniendo solo en cuenta las relaciones directas.

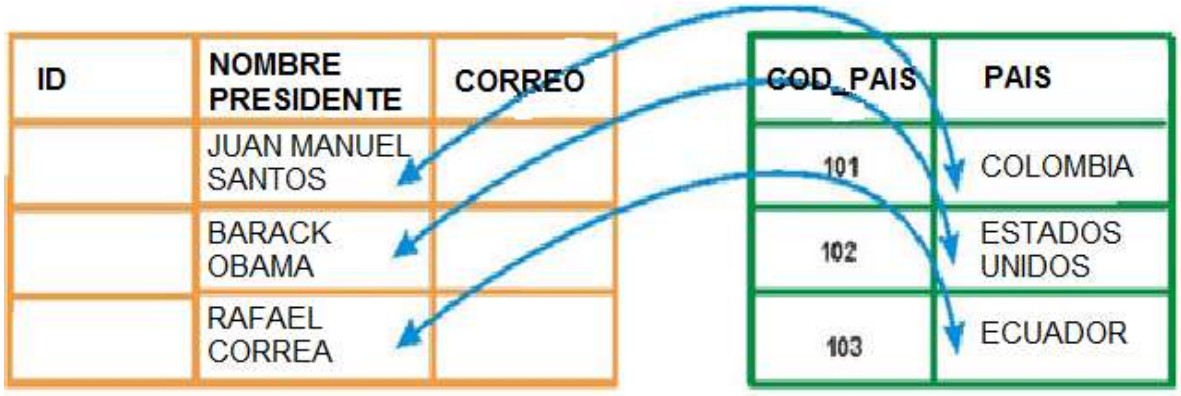
Las relaciones representan asociaciones entre entidades, se denominan el elemento del modelo que permite relacionar en sí los datos. También se denomina como una relación matemática entre varias entidades, cada entidad interviene en una relación con una determinada cardinalidad.

Figura 20. Relaciones representan asociaciones entre entidades

4.1.6. Determinar el orden de la relación. Una vez identificadas todas las relaciones, hay que determinar el grado de la relación, el cual indica el número de elementos que pueden participar en cada uno de los extremos de la relación. Hay tres tipos posibles:

4.1.6.1. Relación uno a uno (1:1). Este tipo de relación se da cuando un registro de una tabla sólo puede estar relacionado con un único registro de otra tabla y viceversa. En este caso, la clave foránea se ubica en alguna de las 2 tablas.

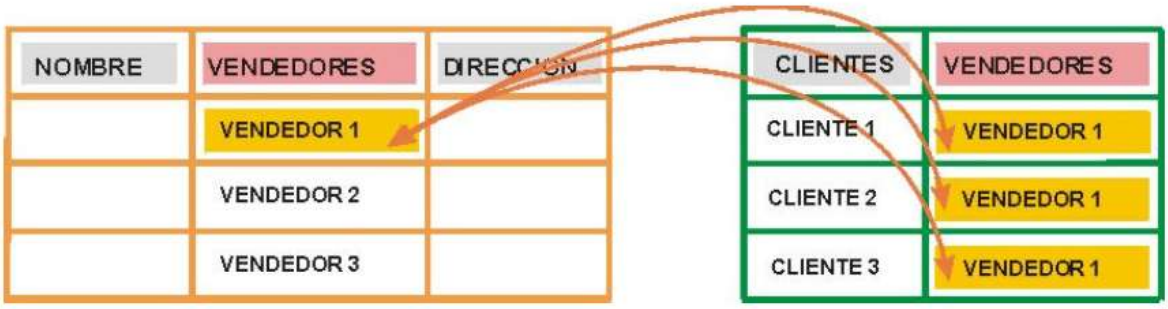
Figura 21. Relación uno a uno (1:1).



Fuente: FRASSIA, Mercedes. Introducción a las bases de datos, un poco de historia. Argentina. 2001. 34 p.

4.1.6.2. Relación uno a muchos (1:n). Este tipo de relación se usa cuando un registro de la tabla (tabla principal) puede estar relacionado con más de un registro de la tabla secundaria. En este caso, se le adiciona a la tabla secundaria, un nuevo atributo, que corresponde a la clave principal de la tabla principal y está se convierte en clave foránea, con la finalidad de proveer un vínculo entre ambas.

Figura 22. Relación uno a muchos (1:n).

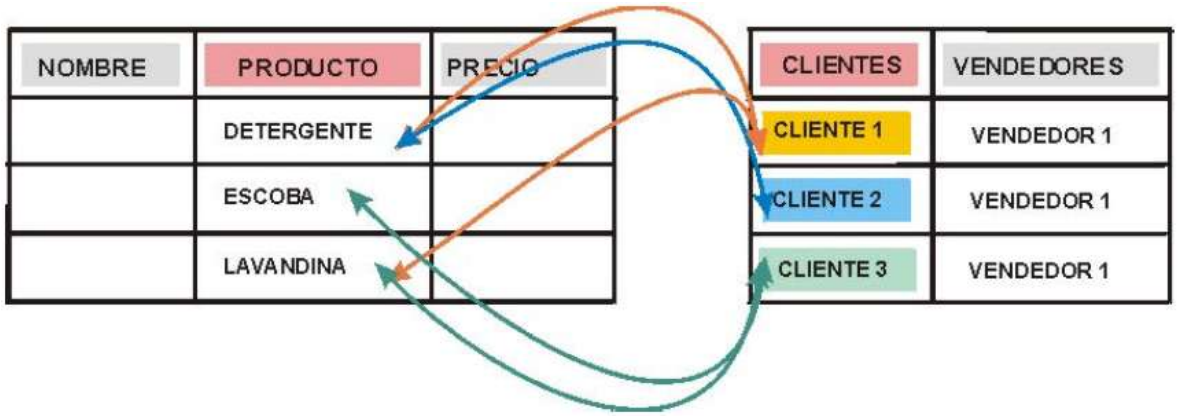


Fuente: FRASSIA, Mercedes. Introducción a las bases de datos, un poco de historia. Argentina. 2001. 34 p.

4.1.6.3 Relación muchos a muchos (n:n). Esta relación se utiliza cuando un registro de una tabla puede estar relacionado con más de un registro de la otra

tabla y viceversa. Para este caso las dos tablas no pueden estar relacionadas directamente, se hace necesario una tabla entre las dos (tabla de vinculación o débil) que incluya los pares de valores relacionados entre sí.

Figura 23. Relación muchos a muchos (n:n).



Fuente: FRASSIA, Mercedes. Introducción a las bases de datos, un poco de historia. Argentina. 2001. 34 p.

4.1.7 Disolver las relaciones. Luego de determinar la cardinalidad de la relación, es necesario disolverlas y de esta forma lograr identificar si existen relaciones muchos a muchos. Las relaciones muchos a muchos no son recomendables y se deben evitar utilizando tablas intermedias que permite dividir la relación muchos a muchos en dos relaciones uno a muchos.

Figura 24. Disolver las relaciones.

Fuente: Relaciones entre tabas. Disponible en internet: http://www.aulapc.es /ofimatica _acces_relaciones.html [Consultado 26 de Enero de 2014].

Este tipo de relación muchos a muchos requiere de una tercera tabla denominada tabla de unión (tabla de vinculación o débil), para la cual es necesario tener en cuenta los siguientes aspectos:

- Los primeros atributos de la nueva tabla corresponden a las claves principales de cada una de las entidades que intervienen en la relación.
- Al mismo tiempo la tabla de vinculación permite incluir los atributos adicionales que permitan definir con mayor propiedad sus características.
- La clave principal de esta nueva tabla es una clave compuesta, donde por lo menos sus dos primeros componentes son las claves principales de las entidades participantes.

- Para finalizar es necesario tener claro las tres tablas están activas, cada una tiene el tamaño correcto para su finalidad y juntas pueden obtener efectos mucho más útiles a nivel informativo de lo que podrían hacer individualmente.

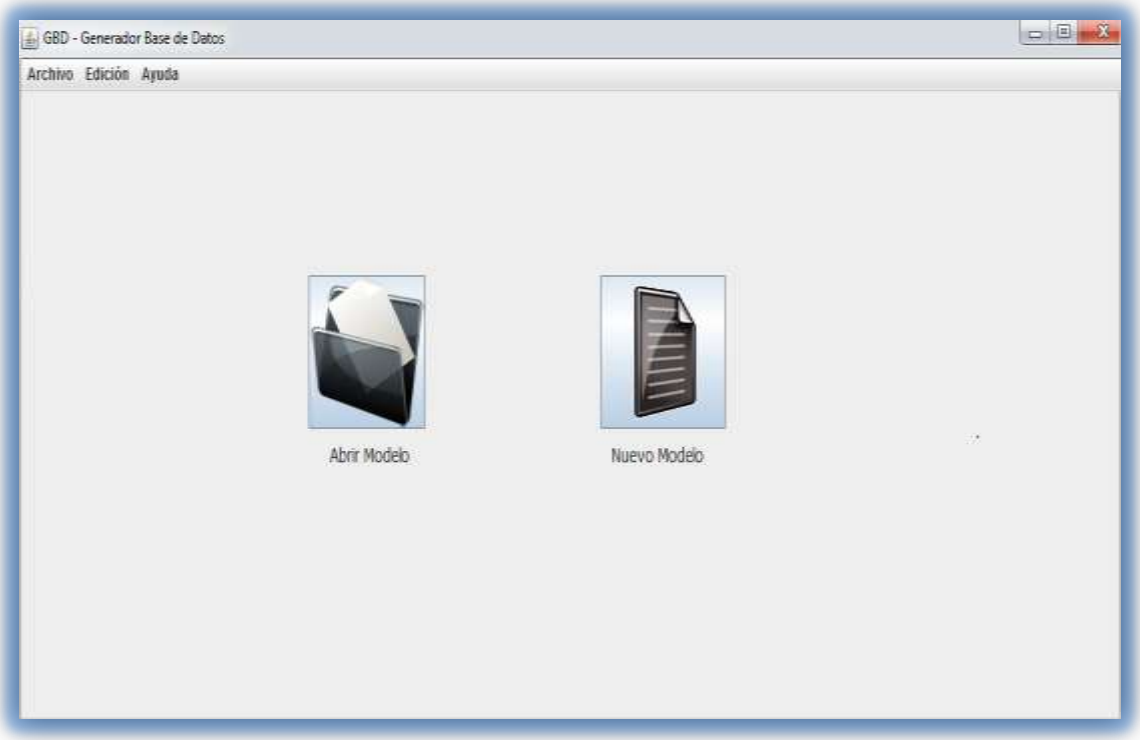
4.1.8, Generación de las sentencias SQL para la creación de la base de datos. Luego de haber definido las características de los objetos que forman parte de la base de datos es necesario generar las sentencias de creación de la base de datos.

4.2. RECORRIDO POR EL SISTEMA

Acorde al planteamiento del proyecto y la ejecución de la metodología Scrum se estructuraron las funcionalidades del software de acuerdo a las necesidades del cliente. Se desarrolló el código fuente de cada uno de los módulos del sistema

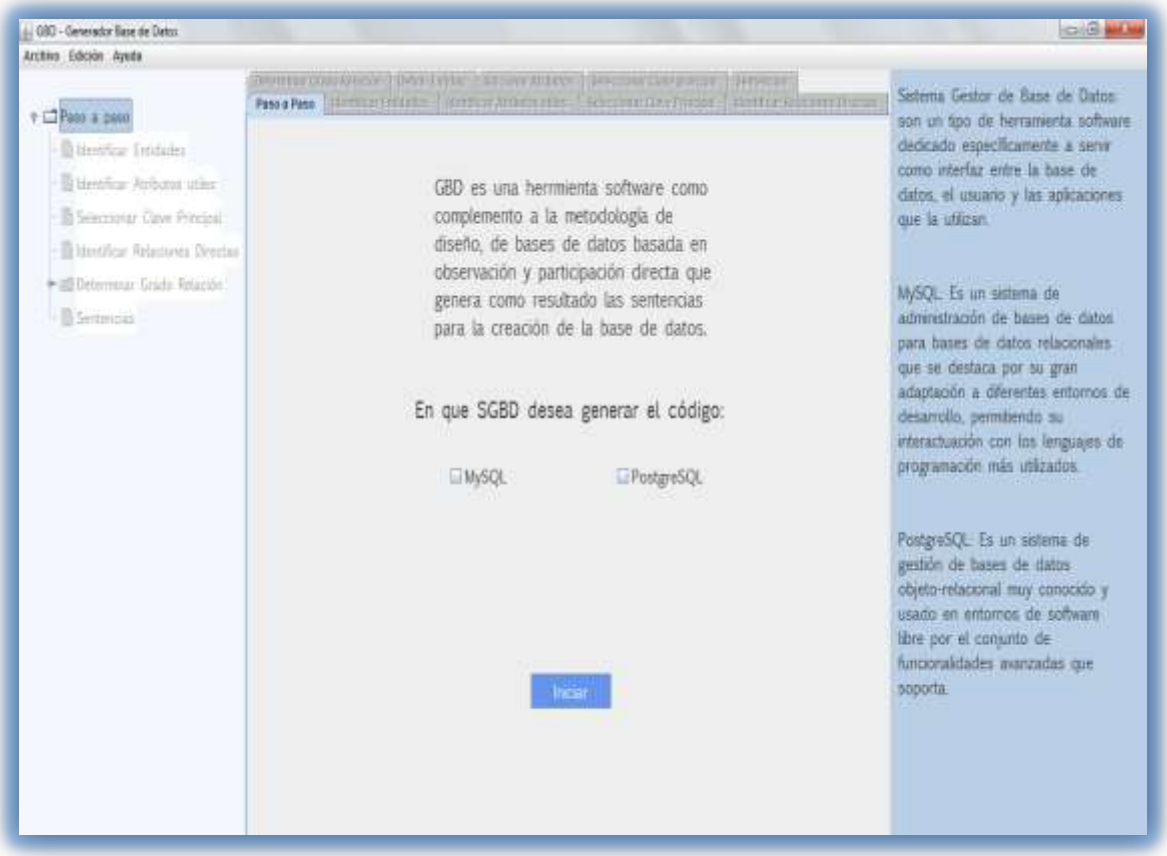
plasmados en cada uno de los sprints, cumpliendo de esta forma con el objetivo principal del proyecto.

Figura 25. Documentación del Sistema y sus módulos



4.2.1. Panel de proyectos. En esta pantalla se encuentra la administración de los proyectos, permite la creación de un nuevo proyecto o abrir un proyecto existente. El usuario puede dar click al botón “Crear Modelo” ubicado en la parte central y de ahí se desplegará la ventana inicial de la herramienta desarrollada. Del mismo modo el usuario puede seleccionar la opción “Abrir Modelo” y de ahí se cargará otra ventana para seleccionar la ubicación del modelo.

4.2.2. Creación de un Nuevo Proyecto



Esta ventana es la primera interfaz que constituye la aplicación de la metodología paso a paso para la creación del modelo de la base de datos. Hay se encuentra en la parte superior izquierda, el árbol de la herramienta, el cual está constituido por los diferentes módulos que se usaran para lograr el modelo de la base de datos. En la parte superior central se encuentra el manejo de las ventanas, conformado por los mismos elementos del árbol de la herramienta. En la parte central se encuentra una breve explicación de lo que es la herramienta y cuál es su finalidad y posteriormente el usuario tendrá la opción de seleccionar el SGBD en el que el usuario desea generar el código para crear la base de datos.

En la parte superior derecha, el usuario encontrará la explicación de cada uno de los términos que irán permitiendo el fácil desarrollo del modelo relacional, los cuales serán explicados a continuación:

Sistema Gestor de Base de Datos: son un conjunto coordinado de programas, procedimientos, lenguajes, herramientas, etc., dedicados específicamente a suministrar los medios necesarios para describir, actualizar y administrar los datos de la BD, estos están compuestos por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta.

MySQL: es un sistema de gestión de bases de datos para bases de datos relacionales que interactúa a través de una aplicación que permite gestionar archivos llamados de bases de datos.

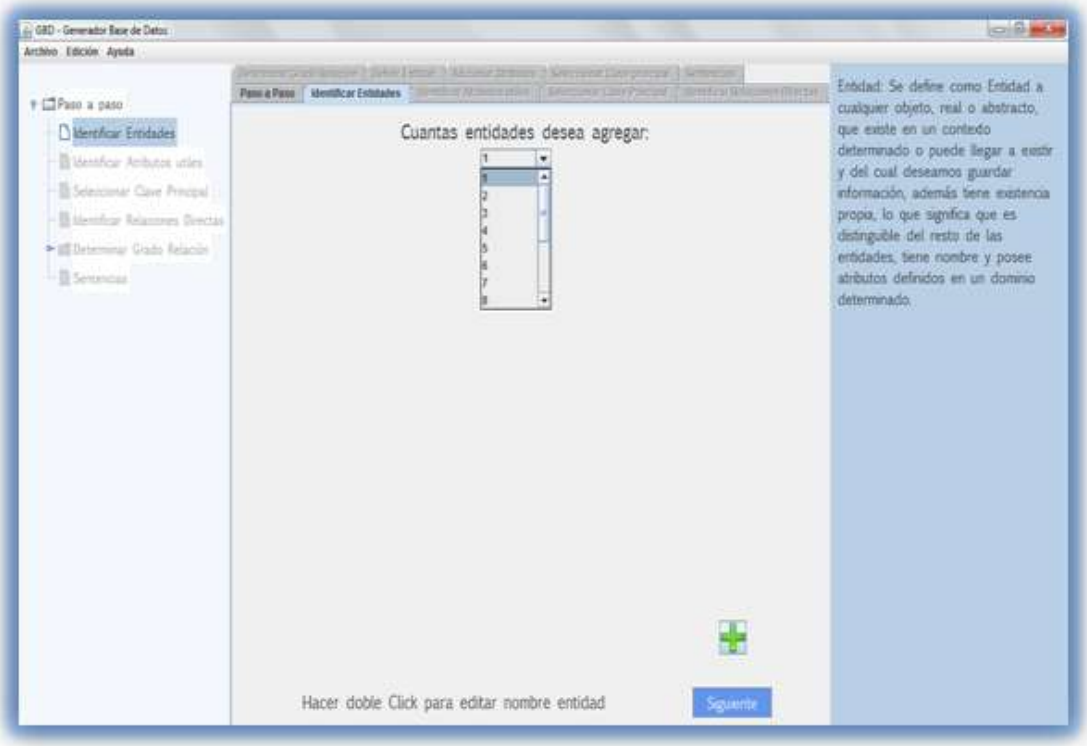
[7]Fue escrito en C y C++ y se destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos, además es muy destacado por la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

PostgreSQL: es un gestor de bases de datos muy conocido y usado en entornos de software libre por el conjunto de funcionalidades avanzadas que soporta, situándolo al mismo o a un mejor nivel que muchos SGBD comerciales.

[7] Que es MySQL. Disponible en Internet: http://www.esepestudio.com/noticias/que-es-mysql. [Consultado 18 de Febrero de 2014]

[8]PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales, ya que utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

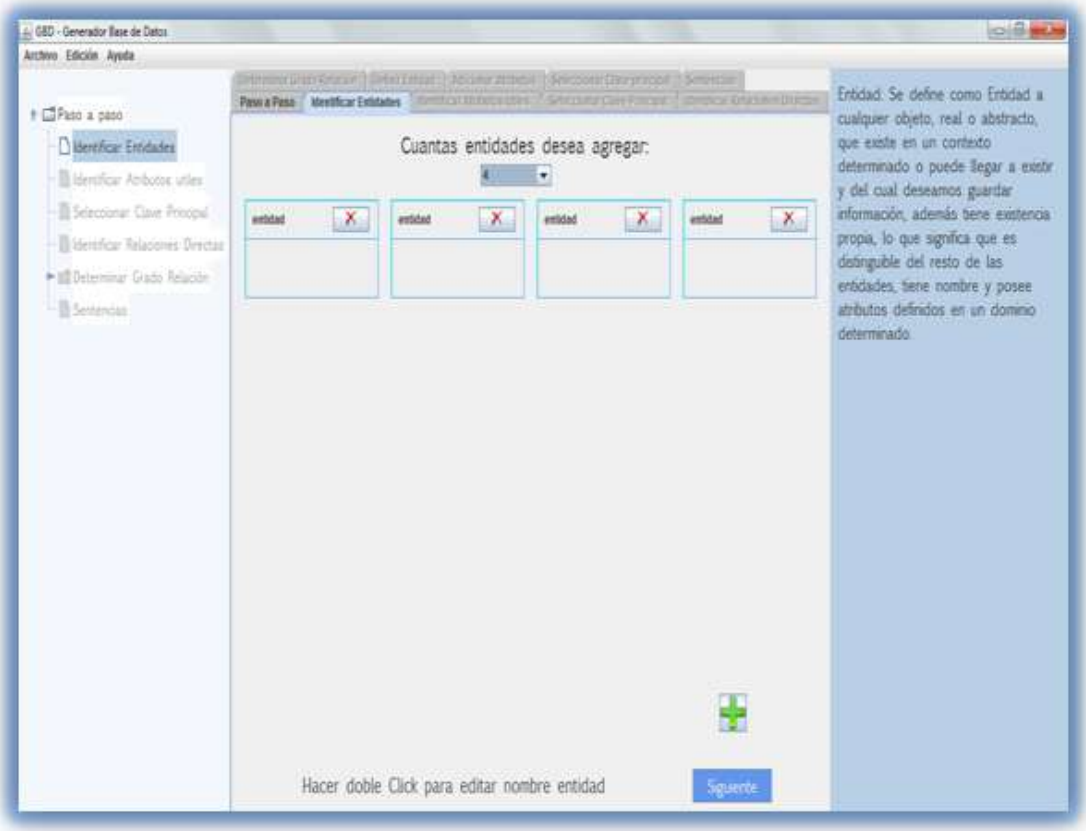
4.2.3. Identificar entidades



Esta pantalla representa el módulo de Gestión de Entidades, el cual permite adicionar las entidades que se consideren necesarias, modificar su nombre o eliminar las entidades ya existentes.

[8] PostgreSQL. [en línea]. Disponible en Internet: http://www.postgresql.org.es/sobre_postgresql. [Consultado 22 de Febrero de 2014]

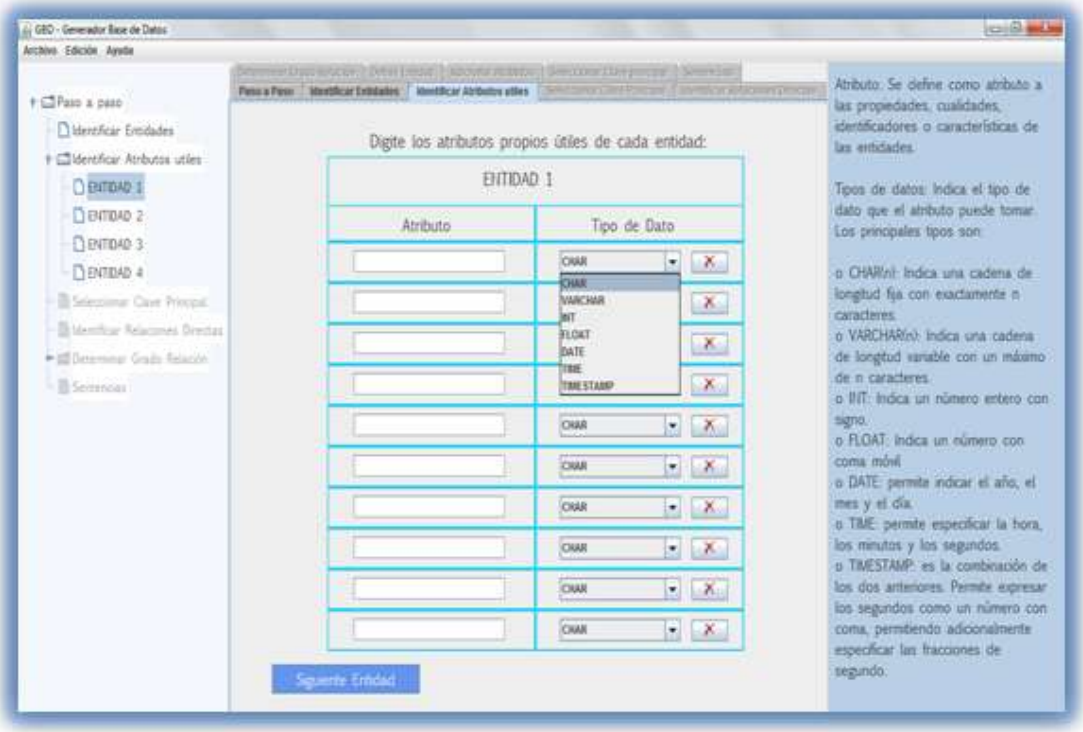
El sistema inicialmente pregunta la cantidad de entidades que el usuario desea agregar y posteriormente el sistema procedera con la creacion de las tablas, desde ahí el usuario puede asignarle el nombre a cada entidad o modificarlo haciendo doble click en el nombre de cada una.



Adicionalmente el usuario podra eliminar las entidades con el botón  y la pregunta de confirmación, asi mismo, en la parte inferior se encuentra el botón  el cual permite adicionar una nueva entidad.

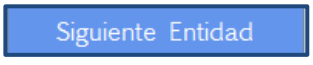
En la parte superior derecha se encuentra la teoria necesaria para el desarrollo de este modulo, la cual ya se definió en la parte inicial de este capítulo.

4.2.4. Identificar Atributos útiles



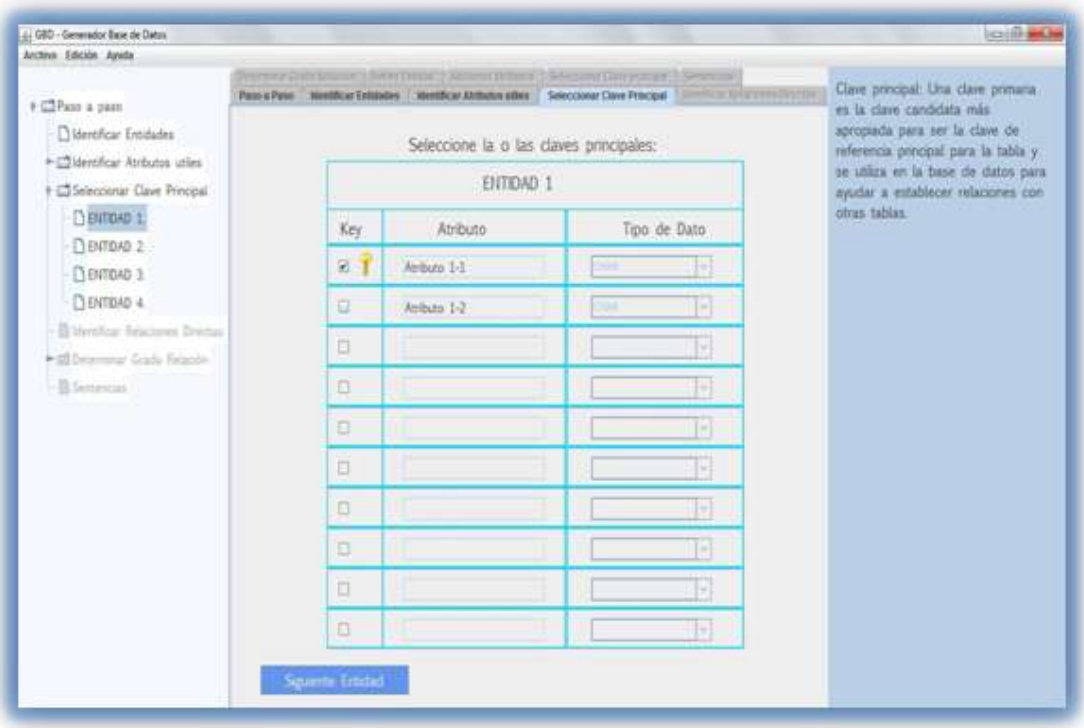
Esta pantalla representa el módulo de Gestión de Atributos, aquí es posible adicionarle atributos a cada entidad, modificar su nombre o tipo de dato y eliminarlos.

La interfaz permite asignarle el nombre a cada atributo y necesariamente seleccionar el tipo de dato de cada uno. Para modificarle el nombre o tipo de dato es necesario hacer doble click sobre el campo al que se desea modificar y para eliminar se debe seleccionar el botón  y aceptar la pregunta de confirmación y de esta forma se eliminará el atributo seleccionado.

En la parte inferior está el botón  el cual permite pasar a la siguiente entidad para la definición de los atributos.

En la parte superior derecha se encuentra la teoria necesaria para el desarrollo de este modulo, la cual ya se definio en la parte inicial de este documento.

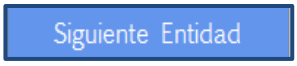
4.2.5. Seleccionar clave principal



Esta pantalla representa el Modulo de Gestión de Claves Principales, aquí es posible seleccionar la(s) claves principales, modificarlas o eliminarlas, según las necesidades del usuario.

Este proceso se desarrolla con una simple selección sobre cuál(es) de los atributos anteriormente definidos, van a ser la(s) claves principales de cada entidad, así mismo, al deseleccionarlas se pueden modificar o eliminar.

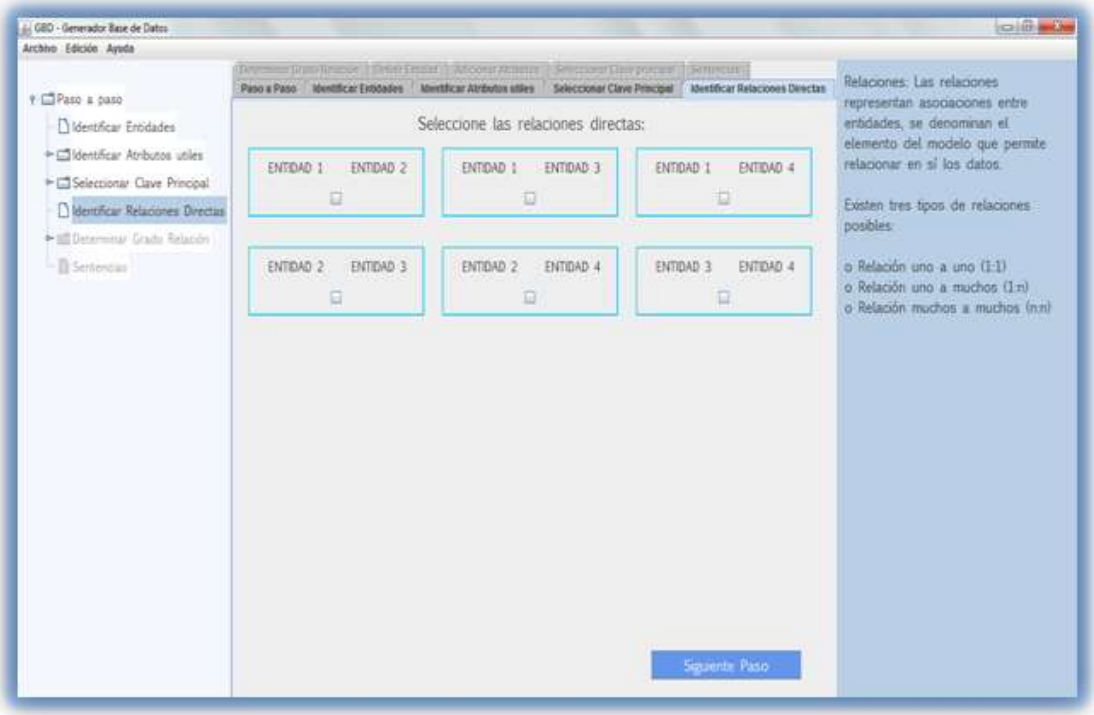
En el árbol del sistema ubicado en la parte izquierda se encuentran todas las entidades y sus atributos creados en el módulo anterior para seleccionarle la(s) claves principales.

En la parte inferior se encuentra el botón 

el cual permite pasar a la siguiente entidad para la selección de la(s) claves principales.

En la parte superior derecha se encuentra la teoria necesaria para el desarrollo de este modulo, la cual ya se definio en la parte inicial de este documento.

4.2.6. Identificar Relaciones Directas.



Esta pantalla constituye el Modulo de Gestión de Relaciones, aquí es posible seleccionar cual(es) duplas de entidades están relacionadas directamente, modificar o eliminar las relaciones.

Esta interfaz consta de todas las posibles duplas de entidades que puedan formarse a partir de las entidades definidas anteriores, por lo tanto, el usuario simplemente selecciona cuales de estas, están relacionadas directamente. Para la

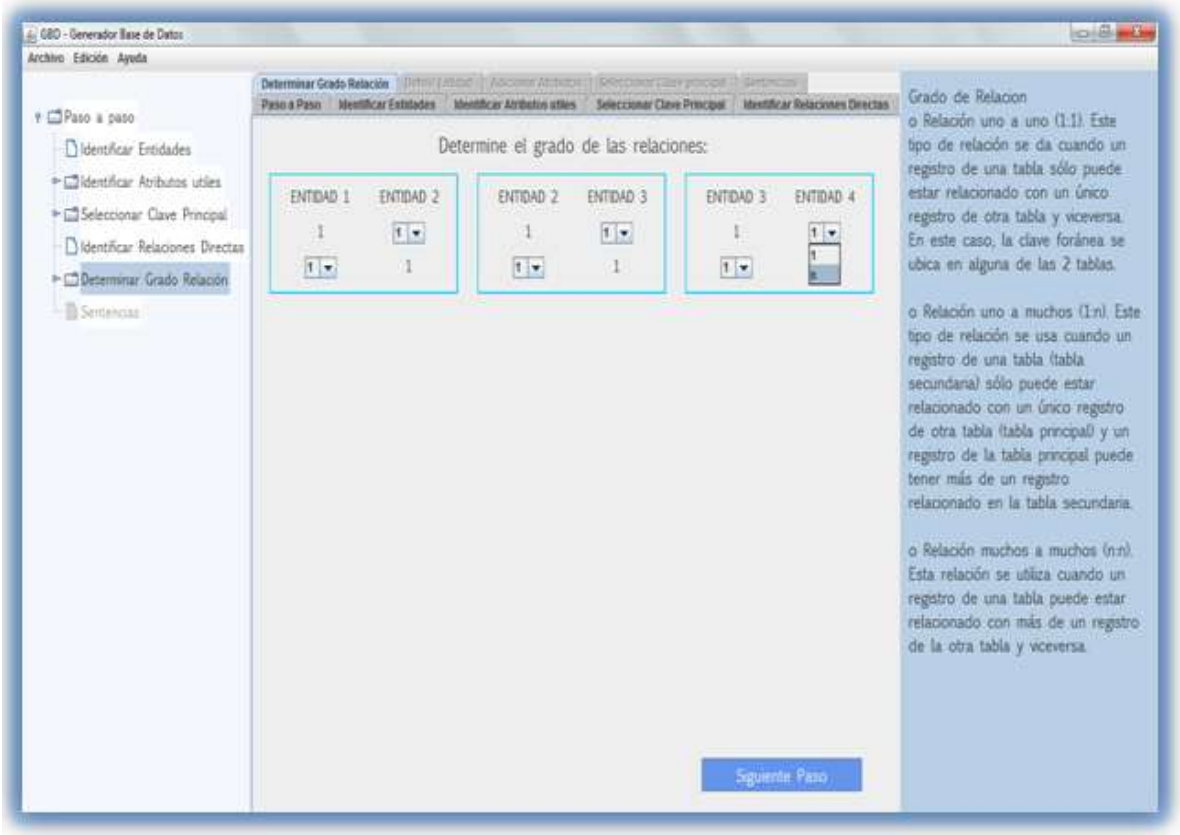
modificación o eliminación de las relaciones se hace un proceso de deselección similar.

En la parte superior derecha se encuentra la teoria necesaria para el desarrollo de este modulo, la cual ya se definio en la parte inicial de este capítulo.

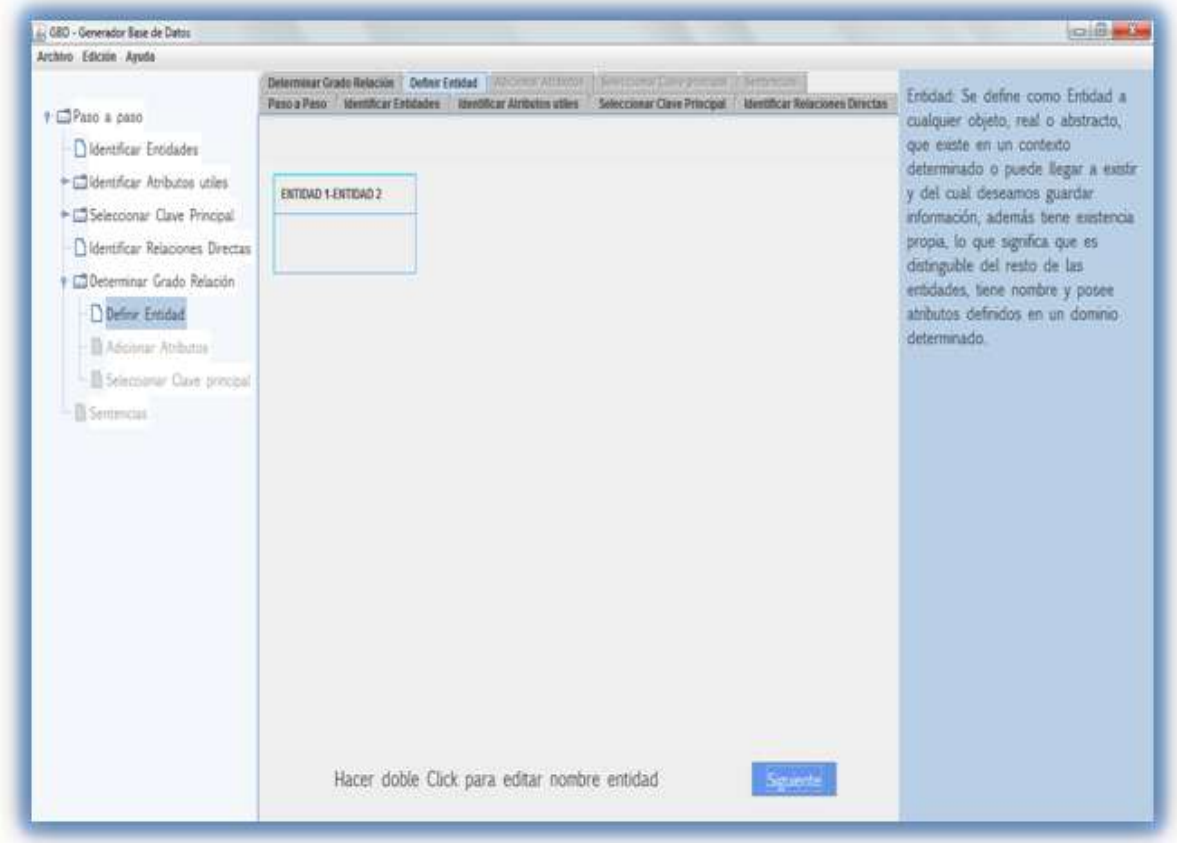
4.2.7. Determinar el Grado de la Relación. Luego de seleccionar cuales entidades están relacionadas, es necesario determinar el grado de las relaciones (uno a uno, uno a muchos o muchos a muchos). Este proceso se realiza a partir de un cuadro en el que aparecen las dos entidades que intervienen en la relación y el usuario debe seleccionar la cardinalidad de la relación con respecto a cada entidad, las opciones son uno (1) o muchos (n), y posteriormente el sistema se encargará de determinar el grado de la relación.

En la parte superior derecha se encuentra la teoria necesaria para el desarrollo de este modulo, la cual ya se definio en la parte inicial de este capítulo.

Si el sistema no detecto ninguna relación muchos a muchos, procederá finalmente a generar las sentencias para crear la base de datos, dependiendo del SGBD seleccionado en la parte inicial de la herramienta. Si por el contrario, el usuario creó una relación muchos a muchos, el sistema automáticamente procederá a crear una nueva entidad y definir todas las especificaciones necesarias para que pueda hacer parte del modelo.

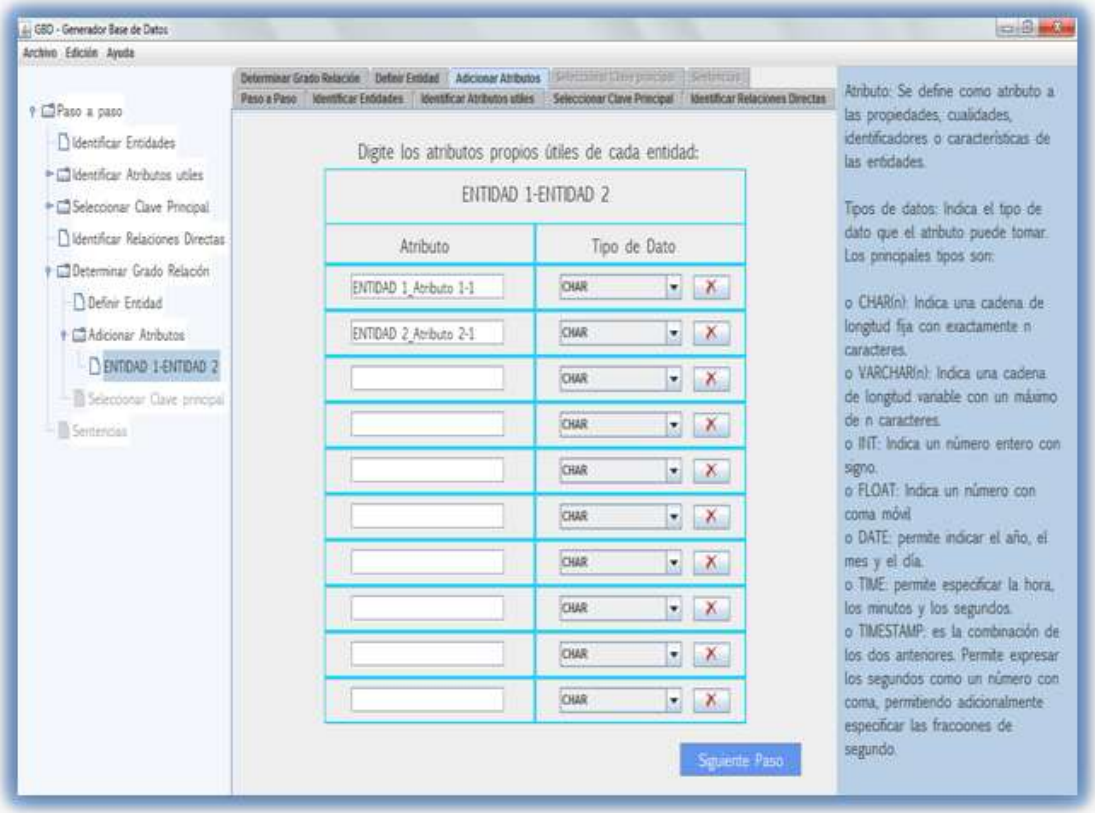


4.2.8. Creación de la entidad auxiliar. En esta interfaz aparece la nueva entidad producto de la relación mucho a muchos. El usuario podrá modificarle su nombre haciendo doble click sobre él.



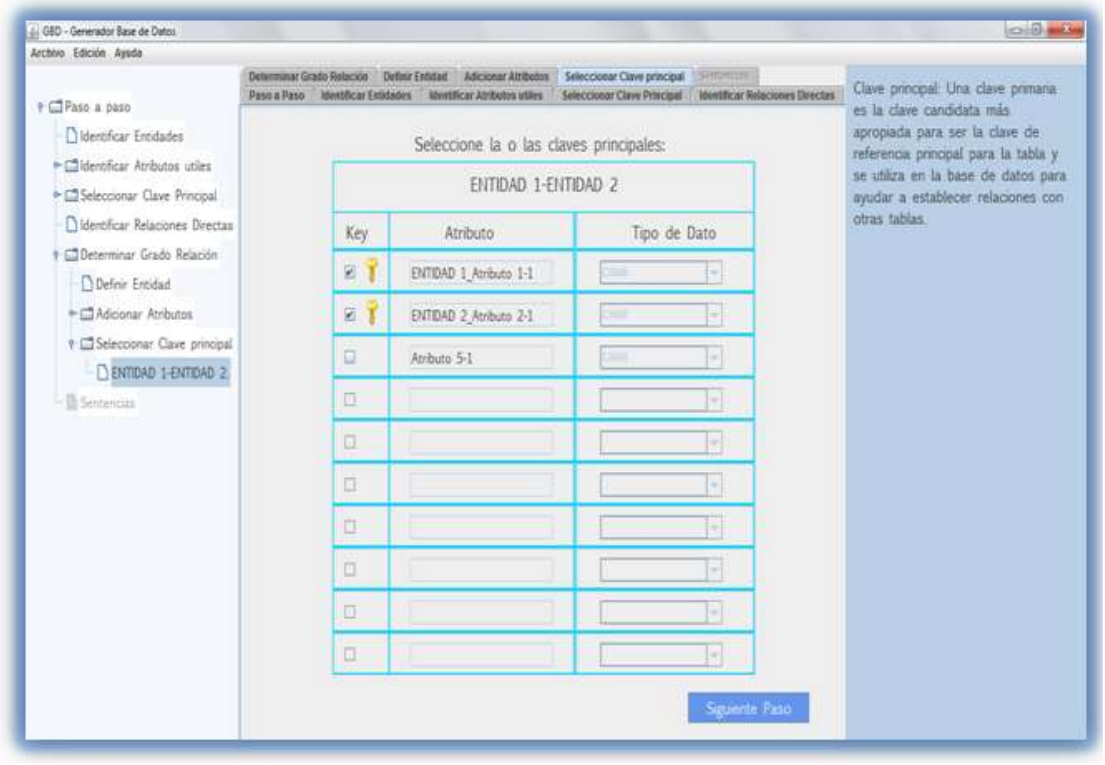
4.2.9. Definición de los atributos de la entidad auxiliar. Esta interfaz permite definir los atributos adicionales que pertenecen a la entidad auxiliar.

Los primeros atributos creados automáticamente por el sistema corresponden a las claves principales de las dos entidades que intervienen en la relación mucho a muchos. El usuario podrá adicionar atributos adicionales a esta entidad, modificarles su nombre o tipo de dato o eliminarlos.



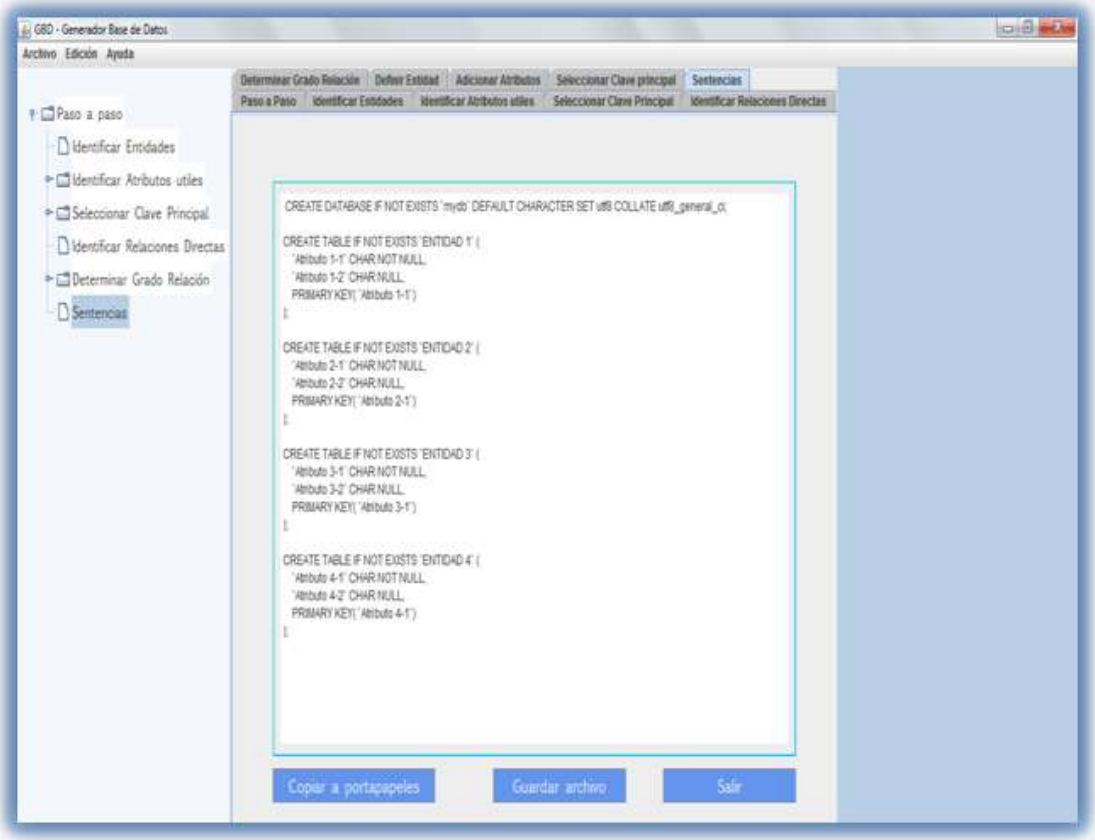
4.2.10. Selección de la(s) claves principales de la entidad auxiliar. Esta interfaz permite seleccionar las claves principales que pertenecen a la entidad auxiliar.

Los primeras claves seleccionadas automáticamente por el sistema son las claves principales de las dos entidades que intervienen en la relación mucho a muchos. El usuario podrá adicionar un nuevo atributo como clave principal, modificarlo o eliminarlo.



4.2.11. Generación de las sentencias MySQL o PosgreSQL para la creación de la Base de Datos. Esta es la última interfaz generara por la herramienta, aquí se visualizaran las sentencias de creación de la base de datos.

El usuario podrá copiar el contenido a portapapeles, guardar el archivo en la ubicación que desee o simplemente salir del programa.


GBD - Generador Base de Datos
Archivo Edición Ayuda
Determinar Grado Relación
Definir Entidad
Adicionar Atributos
Seleccionar Clave principal
Sentencias
Paso a Paso
Identificar Entidades
Identificar Atributos utiles
Seleccionar Clave Principal
Identificar Relaciones Directas
Paso a paso
Identificar Entidades
Identificar Atributos utiles
Seleccionar Clave Principal
Identificar Relaciones Directas
Determinar Grado Relación
Sentencias
CREATE DATABASE IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
CREATE TABLE IF NOT EXISTS `ENTIDAD 1` (
`Atributo 1-1` CHAR NOT NULL,
`Atributo 1-2` CHAR NULL,
PRIMARY KEY(`Atributo 1-1`)
);
CREATE TABLE IF NOT EXISTS `ENTIDAD 2` (
`Atributo 2-1` CHAR NOT NULL,
`Atributo 2-2` CHAR NULL,
PRIMARY KEY(`Atributo 2-1`)
);
CREATE TABLE IF NOT EXISTS `ENTIDAD 3` (
`Atributo 3-1` CHAR NOT NULL,
`Atributo 3-2` CHAR NULL,
PRIMARY KEY(`Atributo 3-1`)
);
CREATE TABLE IF NOT EXISTS `ENTIDAD 4` (
`Atributo 4-1` CHAR NOT NULL,
`Atributo 4-2` CHAR NULL,
PRIMARY KEY(`Atributo 4-1`)
);
Copiar a portapapeles
Guardar archivo
Salir

5. EL LENGUAJE SQL Y LAS SENTENCIAS PARA LA CREACION DE LA BASE DE DATOS

SQL (Structured Query Language), Lenguaje Estructurado de Consulta se define como el lenguaje utilizado para definir, controlar y acceder a los datos almacenados en una base de datos relacional.

SQL como un lenguaje de base de datos, permite crear bases de datos y estructuras de la tabla para realizar tareas de administración de datos básicas, entre ellas esta agregar, eliminar y modificar y así mismo realizar consultas complejas diseñadas para transformar datos sin procesar, en información útil.

SQL satisface muy bien los requerimientos de un lenguaje de base de datos, puesto que sus funciones encajan en dos amplias categorias:

- Un lenguaje de definición de datos que incluye comandos para crear la estructuras de la base de datos y sus elementos
- Un lenguaje de manipulación de datos que incluye comandos para insertar, actualizar, eliminar y recuperar datos dentro de la base de datos.

Entre las grandes ventajas SQL es que realiza funciones básicas con un mínimo esfuerzo por parte del usuario, lo que indica que su estructura y su vocabulario básico de menos de 100 palabras, son fáciles de aprender.

Como ejemplos de sistemas gestores de bases de datos que utilizan SQL podemos citar DB2, SQL Server, Oracle, MySql, Sybase, PostgreSQL o Access.
El SQL es un lenguaje universal que se emplea en cualquier sistema gestor de bases de datos relacional, puede ser SQL Server, PostgreSQL, MySQL, Oracle,

entre otros, el cual tiene un estándar definido a partir del cual cada sistema gestor ha desarrollado su versión propia.

El lenguaje SQL está compuesto por comandos, operadores, cláusulas, y funciones de agregado. Dichos elementos se combinan en las instrucciones y permiten crear, actualizar y manipular las bases de datos.

5.1 HISTORIA DE SQL

SQL (Structured Query Language) inicio en 1974 con su definición “lenguaje para especificar las características de las bases de datos que adoptaban el modelo relacional” dada por Donald Chamberlin, en compañías de sus colaboradores que trabajaban en los laboratorios de investigación de IBM.

Se implementó en un prototipo llamado SEQUEL-XRM entre 1974 y 1975. Luego de experimentar con este prototipo condujeron entre 1976 y 1977 a una revisión del lenguaje (SEQUEL/2), que a partir de ese momento cambió de nombre por motivos legales, convirtiéndose en SQL. El prototipo (System R), basado en este lenguaje, se adoptó y utilizó internamente en IBM, el cual fue adoptado por algunos de sus clientes elegidos.

Gracias al éxito de este sistema y a pesar que aún no estaba comercializado, otras compañías empezaron a desarrollar sus productos relacionales basados en SQL. EN 1981, IBM empezó a entregar sus productos relacionales y en 1983 empezó a vender DB2.
En los años ochenta, numerosas compañías (Oracle y Sybase, entre otras) empezaron a comercializar sus productos basados en SQL, convirtiéndose así en el estándar industrial enfocado directamente a las bases de datos relacionales.

En 1986, ANSI adoptó SQL como estándar para los lenguajes relacionales y en 1987 se convirtió en estándar ISO. Esta versión del estándar fue nombrada como SQL/86. En los años posteriores se generaron diversas revisiones que han conllevaron a la versión SQL/89 y, posteriormente, a la actual SQL/92.

Lograr estandarizar un lenguaje para las bases de datos relacionales abrio el camino a la intercomunicabilidad entre todos los productos que se basan en él.

Hoy en día, cada productor adopta e implementa en su base de datos únicamente el corazón del lenguaje SQL (Entry level o en ocasiones el Intermediate level), extendiéndose de manera individual a sus necesidades y visión propia del mundo de las bases de datos.

Actualmente, está en marcha un proceso de revisión del lenguaje por parte de los comités ANSI e ISO, que debe concluir con la definición de lo que en este momento se conoce como SQL3, y la transformación de SQL en un lenguaje stand-alone (mientras ahora se usa como lenguaje hospedado en otros lenguajes) y la introducción de nuevos tipos de datos más complejos que permitan, por ejemplo, el tratamiento de datos multimediales.

5.2 EL LENGUAJE SQL Y LA CREACIÓN DE LA BASE DE DATOS

El Lenguaje de Definición de Datos (LDD), es la parte de SQL dedicada a la definición de la base de datos y el nivel interno. Este consta de sentencias para definir la estructura de la base de datos, permitiendo con ellas, crear la base de datos, crear, modificar o eliminar la estructura de las tablas, crear índices, definir reglas de validación de datos, relaciones entre las tablas, etc.

Existen tres sentencias que se emplean en SQL para crear o modificar una base de datos. Estas sentencias son:

- CREATE, que define y crea un objeto en la base de datos.
- DROP, que elimina un objeto existente en la base de datos.
- ALTER, que modifica la definición de un objeto de la base de datos.

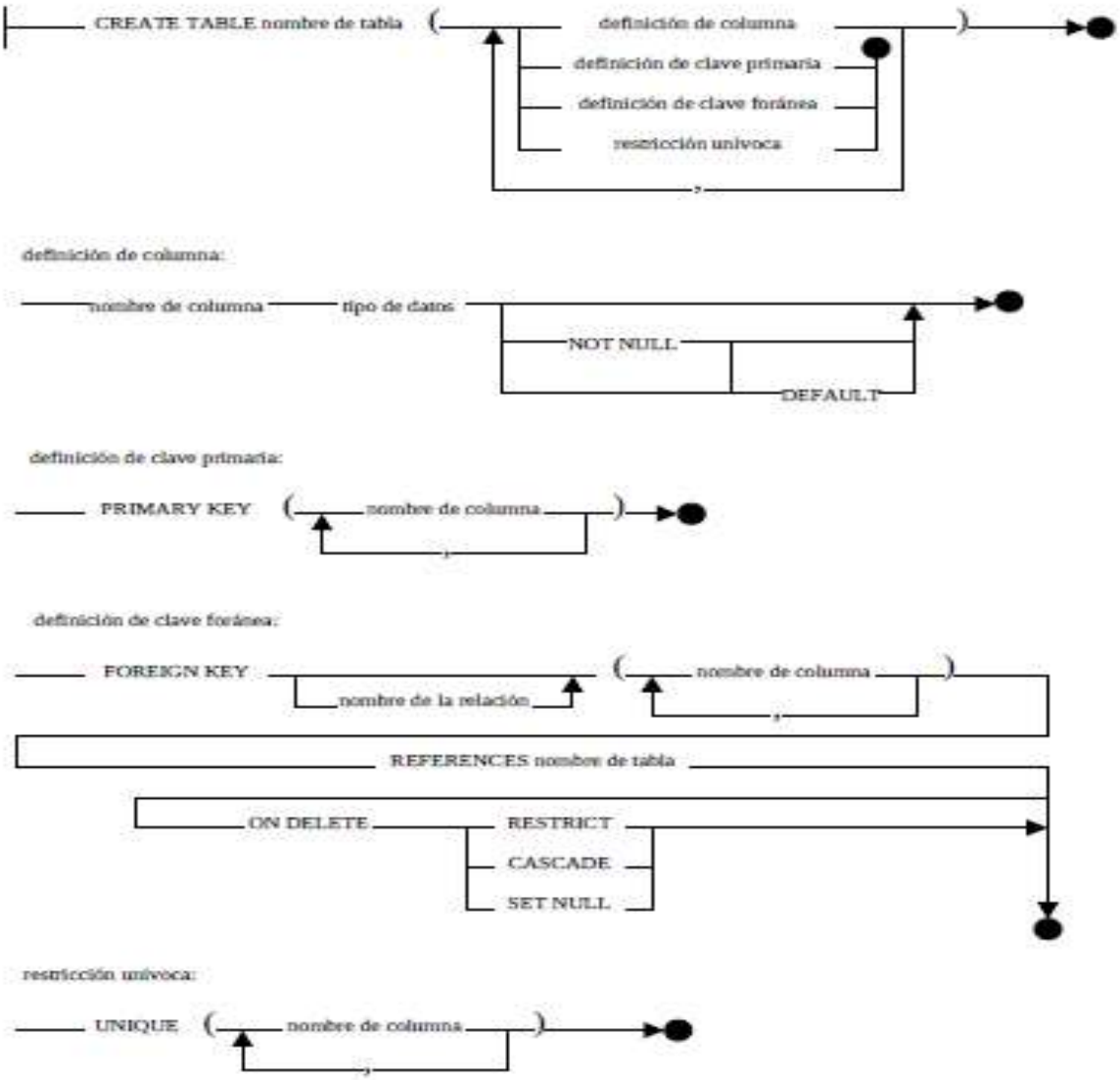
5.2.1 Sentencia CREATE. La sintaxis empleada por SQL para la creación de la base de datos es la siguiente:

CREATE DATABASE nombre_base de datos

Una vez creada la base de datos, se pueden crear las tablas que la componen. La tabla es la estructura más importante de una base de datos relacional. Estas son creadas por el administrador de la base de datos y utilizadas posteriormente por los usuarios. La instrucción SQL propuesta para este fin es:

CREATE TABLE nombre_tabla ()

Figura 26.Diagrama sintáctico de la sentencia Create Table



Fuente: El lenguaje SQL III: Creación de bases de datos y seguridad. Disponible en internet: http://informatica.uv.es/estguia/ATD/apuntes/teoria/documentos/SQL-III.pdf [Consultado 24 de Enero de 2014].

La creación de una tabla conlleva como se muestra en el diagrama sintáctico, la definición de otros elementos fundamentales que conforman la base de datos, entre ellos se encuentran:

- Columnas:

Se definen como el cuerpo de la sentencia CREATE TABLE y estas aparecen en una lista separada por comas e insertada entre paréntesis.

```
CREATE nombre_tabla (
        nombre_columna  tipo_de_dato
);
```

El orden descrito en la sentencia, determina el orden de izquierda a derecha de las columnas en la tabla.

Cada columna posee las siguientes características:

- Cada columna de la tabla debe tener un nombre único, pero los nombres pueden ser iguales a los de las columnas de otras tablas. Los nombres deben comenzar con un carácter alfabético y es recomendable usar nombres no tan extensos dado que SQL Entry Level prevé nombres no mayores de 18 caracteres.

- El tipo de datos de la columna identifica la clase de datos que la columna puede almacenar.

- La columna puede o no contener datos nulos, la cláusula NOT NULL impide que aparezcan valores NULL en la columna.

- La columna puede contener un valor por omisión.

- Tipo de datos de la columna:

Indica el tipo de dato que la columna podrá contener. Los principales tipos previstos por el estándar SQL son:

- *CHARACTER(n):* Indica una cadena de longitud fija con exactamente n caracteres. *CHARACTER* se puede abreviar con *CHAR*.

- *CHARACTER VARYING(n):* Indica una cadena de longitud variable con un máximo de n caracteres. *CHARACTER VARYING* se puede abreviar con *VARCHAR* o *CHAR VARYING*.

- *INTEGER*: Indica un número entero con signo. La precisión (tamaño del número entero), depende de la implementación del Sistema Gestor que esté siendo utilizado. Se puede abreviar con *INT*.

- *SMALLINT*: Indica un número entero con signo y una precisión (tamaño) no superior a *INTEGER*.

- *FLOAT(p)*: Indica un número con coma móvil y una precisión p. El valor de la precisión depende del sistema gestor que se esté utilizando. También es posible usar *FLOAT* sin indicar la precisión, es decir, con la precisión por defecto.

- *DECIMAL(p,q)*: Indica un número con coma fija de por lo menos p cifras y signo, con q cifras después de la coma. El valor de p depende de la implementación. Se puede abreviar con *DEC*.

- *INTERVAL*: Indica un periodo de tiempo (años, meses, días, horas, minutos, segundos y fracciones de segundo).

- *DATE, TIME* y *TIMESTAMP*: Indica un instante de tiempo preciso.

- *DATE*: permite indicar el año, el mes y el día.
- *TIME*: permite especificar la hora, los minutos y los segundos.
- *TIMESTAMP*: es la combinación de los dos anteriores. Permite expresar los segundos como un número con coma, permitiendo adicionalmente especificar las fracciones de segundo.

- Valores por defecto:

La cláusula DEFAULT es la encargada de asignarle un valor por defecto a un campo a través de una definición de campo en la sentencia CREATE TABLE.
La sintaxis que se usa es la siguiente:

DEFAULT { valor | NULL }

donde valor es un valor válido para el tipo de dato con el que la columna se ha definido o NULL que indica que el campo puede contener valores nulos.

```
CREATE TABLE video_sales (
    did      VARCHAR(40) DEFAULT 'luso films',
    number   INTEGER DEFAULT 0,
    total    CASH DEFAULT '$0.0'
);
```

- Restricciones: Una restricción es una cláusula que obliga al cumplimiento de ciertas limitaciones en la base de datos. Están se consideran como un método que permite la implementación de reglas y cumplen la función de restringir los datos que pueden ser almacenados en las tablas. Además se cree que no son parte formal del modelo relacional, pero son incluidas porque tienen la labor de organizar los datos de manera más eficiente.

El lenguaje SQL posibilita los siguientes tipos de restricciones que se especifican durante la creación de las tablas:

- Clave primaria: La restricción de campo PRIMARY KEY específica un campo de una tabla que solamente puede contener valores únicos y no nulos, es decir, en este campo no se permiten valores duplicados ni nulos.

  Esta clave primaria define unívocamente a todos los demás atributos de la tabla, para lograr especificar los datos que luego serán relacionados con las demás tablas. Es importante recordar que sólo puede existir una única clave primaria por tabla.

```
CREATE TABLE estudiantes (
   código     INT(7) PRIMARY KEY,
   nombre     VARCHAR(40) UNIQUE,
   carrera    VARCHAR(30)
);
```

- Clave foránea: La restricción FOREIGN KEY establece las relaciones entre dos tablas, esto se logra cuando las columnas de una de las tablas hacen referencia a las columnas de la otra, que contienen el valor de clave principal. Esta columna se convierte en una clave externa para la segunda tabla.

  La clave foránea se define como una columna o combinación de columnas que son utilizadas para determinar vínculo entre los datos de dos tablas.

Las cláusulas ON DELETE y ON UPDATE indican qué acción hay que ejecutar en el caso en que una columna en la tabla referenciada sea eliminada o actualizada. Las acciones pueden ser:

- NO ACTION: especifica que la eliminación produce un error.

- CASCADE: especifica que se eliminaran todas las filas con claves externas que apuntan a la fila eliminada.

- SET DEFAULT: especifica que todas las filas con claves externas que apuntan a la fila eliminada se establecen en sus valores predeterminados.

- SET NULL: especifica que todas las filas con claves externas que apuntan a la fila eliminada se establecerán en NULL.

```
CREATE TABLE escuela (
    código     INT(7) PRIMARY KEY,
    nombre     VARCHAR(40) UNIQUE,
    facultad     VARCHAR(20)
);

CREATE TABLE estudiantes (
    código     INT(7) PRIMARY KEY,
    nombre     VARCHAR(40) UNIQUE,
    escuela_codigo     VARCHAR(30)
     FOREIGN KEY REFERENCES escuela (código)
   ON DELETE SET NULL,
);
```

- Not Null: La restricción NOT NULL establece una ley que no permite que un campo contenga valores nulos. Es importante tener en cuenta, que ésta restricción funciona únicamente como una restricción de campo y como restricción de tabla, no está permitida.

```
CREATE TABLE estudiantes (
    código     INT(7) NOT NULL,
    nombre     VARCHAR(40) NOT NULL,
    carrera    VARCHAR(30)
);
```

- Unique: La restricción UNIQUE establece una ley que exige que un grupo de uno o más campos de una tabla deba contener valores únicos, por lo tanto, no están permitidos valores repetidos en un atributo.

```
CREATE TABLE estudiantes (
    código     INT(7) NOT NULL,
    nombre     VARCHAR(40) UNIQUE,
    carrera    VARCHAR(30)
    );
```

- Check: La restricción CHECK garantiza que todos los valores de una columna cumplan ciertas condiciones, esta especifica los valores que están permitidos en un campo o en una tabla, lo que declara a la restricción CHECK como una restricción de campo y de tabla.

  Las restricciones de tabla se declaran de forma independiente de la definición de las columnas y se pueden aplicar a varias columnas de la tabla.

```
CREATE TABLE estudiantes (
```

```
    código     INT(7) NOT NULL,
    nombre     VARCHAR(40) UNIQUE,
    carrera    VARCHAR(30)
    activo     CHAR(2)  CHECK (activo ='si' or activo ='no')
);
```

5.2.2. Sentencia DROP. Una base de datos, en su intención de representar la realidad, debe sufrir modificaciones a lo largo de su existencia, para ellos se hace necesario añadir nuevas tablas u eliminar otras tablas que ya no sean necesarias.

El comando DROP de SQL es utilizado para eliminar un objeto de la base de datos. Es posible la eliminación de toda la base de datos a partir del siguiente comando:

```
DROP DATABASE nombre_base_de_datos;
```

También es posible eliminar una tabla, si está se elimina, todas las filas de la tabla serán borradas y se quitará la tabla de la estructura de la base de datos; una vez que se elimina una tabla, ya no se puede recuperar, por lo tanto, todos los contenidos se pierden y no existe forma de recuperar los datos, además de que todas las referencias hechas a esa tabla no serán válidas.

La sintaxis utilizada es:

```
DROP TABLE "nombre_tabla";
```

5.2.3. Sentencia ALTER. Una vez que se crea la tabla en la base de datos, los usuarios suelen manifestar la necesidad de almacenar información adicional con respecto a las entidades representadas en las tablas, por ello se hace indispensable poder modificar la estructura de la tabla.

La sentencia ALTER permite variar la definición de una tabla ya creada, más específicamente admite modificar la definición de columnas ya existentes (MODIFY, ALTER), añadir más columnas o restricciones (ADD), eliminar columnas y restricciones (DROP) y habilitar/deshabilitar restricciones.

- Adición de una columna: El uso más común de la sentencia ALTER TABLE es para añadir una columna a una tabla existente. La cláusula de definición de la columna en la sentencia ALTER TABLE de modo que la nueva columna será añadida al final de las definiciones de columnas de la tabla y aparecerá como la columna más a la derecha en consultas posteriores.
  Por defecto, el gestor de la base de datos supone un valor NULL para la columna recién añadida en todas las filas existentes en la tabla.

La sintaxis utilizada para adicionar una columna es la siguiente:

ALTER TABLE nombre_tabla ADD nombre_columna tipo_de_dato;

- Eliminación de una columna:

Otro uso de la sentencia ALTER TABLE es para eliminar una columna ya existente y SQL lo permite utilizando la sentencia:

ALTER TABLE nombre_tabla DROP nombre_columna;

Cuando se elimina una columna de una tabla, dicha columna y todos los datos que contiene se eliminan de la base de datos, por lo tanto, es necesario tener mucha atención dado que esta acción no se puede deshacer.

También es necesario tener presente que antes de eliminar una columna que contenga una restricción, es necesario eliminar la restricción como primer paso.

- Modificación del tipo de datos de una columna: La sentencia ALTER permite adicionalmente modificar cualquier elemento de la base de datos. Por ejemplo es posible eliminar el tipo de datos de la columna de la siguiente manera:

ALTER TABLE nombre_tabla MODIFY nombre_columna tipo_de_dato;

Otro uso habitual de ALTER TABLE es cambiar o añadir definiciones de los diferentes elementos de una tabla. Las sentencias que añaden, eliminan o modifican definiciones de la base de datos son diversas, tal y como puede verse en los siguientes ejemplos:

- Adición de la clave primaria:

ALTER TABLE nombre_tabla ADD PRIMARY KEY nombre_columna;

- Eliminación de la clave primaria:

ALTER TABLE nombre_tabla DROP PRIMARY KEY;

- Adición de una clave foránea:

ALTER TABLE ADD FOREIGN KEY (nombre_columna) REFERENCES nombre_tabla_padre(nombre_clave_primaria);

- Eliminación de una clave foránea:

ALTER TABLE nombre_tabla DROP FOREIGN KEY nombre_columna;

- Adición de una restricción UNIQUE:

ALTER TABLE nombre_tabla ADD UNIQUE nombre_columna;

- Eliminación de una restricción UNIQUE:

ALTER TABLE nombre_tabla DROP UNIQUE nombre_columna;

- Adición de una restricción CHECK:

ALTER TABLE nombre_tabla ADD CHECK (nombre_columna [condición]);

- Modificación de un valor por omisión:

ALTER TABLE nombre_tabla ALTER nombre_columna DEFAULT { valor | NULL };

- Eliminación de un valor por omisión:

ALTER TABLE nombre_tabla ALTER nombre_columna DROP DEFAULT;

- Sintaxis para cambiar el nombre de una tabla.

El comando RENAME SQL es utilizado para cambiar el nombre de la tabla o un objeto de base de datos. Si cambia el nombre del objeto cualquier referencia al nombre antiguo se verá afectada, por lo que es necesario cambiar manualmente el nombre antiguo por el nuevo nombre en cada referencia.

RENAME antiguo_nombre_tabla To nuevo_nombre_tabla;

6. CONCLUSIONES

Luego de culminado el proyecto y tras la implementación de la metodología Scrum, se logró desarrollar una herramienta software que a partir de una serie de pasos muy didácticos permite crear el modelo final de base de datos y genere las sentencias de creación de la base de datos.

El proyecto cumplió los objetivos propuestos y se desarrolló durante el tiempo estimado, basándose principalmente en investigaciones previas permitiéndonos realizar las siguientes afirmaciones:

- Scrum es un método de gestión de proyectos en general que se está utilizando con gran éxito en el área de desarrollo de software. Los resultados obtenidos en la ejecución del presente proyecto permitieron evidenciar su utilidad y validez, especialmente en proyectos pequeños y medianos, con grupos de trabajo pequeños y que involucran constantemente al dueño del producto.

- La comunicación constante entre los implicados del proyecto constituyó un pilar fundamental en la conclusión exitosa del mismo. Scrum indica que es necesario realizar reuniones continúas que permitan evaluar los avances, las taras a realizar y el progreso de cada sprint, tanto de gestión como de codificación del proyecto y si es necesario, aplicar los correctivos correspondientes.

- El modelo de bases de datos relacional es el estándar de las bases de datos actuales estableciéndose como el principal modelo de datos, debido a su

simplicidad, que facilita el trabajo del programador, proporcionando una forma muy simple y potente de representar datos.

- El lenguaje SQL ha crecido durante las dos últimas décadas desde un lenguaje simple con pocas características a un lenguaje complejo para satisfacer a los diferentes tipos de usuarios. Este posee grandes ventajas como su amplio uso, su idioma completo que implementa todas las operaciones del modelo relacional, su idioma sencillo con menos de 30 comandos, además de una curva de aprendizaje realmente rápida, lo que indica que el usuario no necesita ser programador para poder usarlo.

- El uso de Java como lenguaje base en el desarrollo del proyecto cumplió con el objetivo de realizar una codificación rápida y acorde al cronograma establecido. Se logró implementar una herramienta que apoye a estudiantes y demás aprendices sobre el diseño de bases de datos relacionales a partir de una sencilla serie de pasos que permiten desarrollar un modelo estable de la base de datos y que genere como resultado, las sentencias de creación de la base de datos, ya sea en MySQL o en PostgreSQL.

7. RECOMENDACIONES

- Debido a que el proyecto es un prototipo, se recomienda continuar con nuevas versiones que implementen nuevas necesidades del cliente y del entorno, por ello se recomienda implementar el esquema relacional de la base de datos, este sería un gran aporte y permitiría que el usuario obtuviera el diagrama relacional de la base de datos como resultado adicional de la herramienta.

- También sería útil ampliar el alcance del software, para ellos se recomienda generar las sentencias de creación de la base de datos en todos los SGDB que existen, no solamente en MySQL y PostgreSQL.

BIBLIOGRAFIA

AMAYA, Jairo. Sistemas de información gerenciales. Segunda edición. Bogota: ECOE Ediciones, 2009. 228 p.

ANGARITA, Heiner y TOSCANO, Luis Gabriel. Prototipo de sistema de información web para la gestión de proyectos software bajo la metodología Scrum. Universidad Industrial de Santander. 2013. 85 p.

Bases de Datos. Disponible en internet: http://www.lsi.us.es/docencia/get.php?id=1456 [Consultado 15 de Enero de 2014].

CORONEL, Carlos; MORRIS, Steven y ROB, Peter. Bases de Datos, Diseño, Implementación y Administración. Novena edición. México: Cengage Learning Editores, 2011.250 p.

CUADRA, Dolores; CASTRO, Elena y MARTÍNEZ, Paloma. Desarrollo de Bases de Datos: casos prácticos desde el análisis a la implementación. España: RA-MA S.A. Editorial y Publicaciones, 2007. 569 p

El lenguaje SQL III: Creación de bases de datos y seguridad. Disponible en internet: http://informatica.uv.es/estguia/ATD/apuntes/teoria/documentos/SQL-III.pdf [Consultado 24 de Enero de 2014].

ELMASRI, Ramez y NAVATHE, Shamkant B. Sistemas de Bases de Datos: CONCEPTOS FUNDAMENTALES. Segunda Edición. Ed. Addison-Wesley Iberoamericana, 1997. 887 p.

FRASSIA, Mercedes. Introducción a las bases de datos, un poco de historia. Argentina. 2001. 34 p.

MySQL. Disponible en internet: http://www.mysql.com [Consultado 12 de Febrero de 2014].

PONS, Olga; MARÍN, Nicolás; MEDINA, Juan Miguel; ACID, Silvia y VILA, Mª Amparo. Introducción a las Bases de Datos: El modelo relacional. Madrid: Thomson Editores, 2005. 281 p.

PostgreSQL. Disponible en internet: http://www.postgresql.org.es [Consultado 15 de Febrero de 2014].

ROB, Peter y CORONEL, Carlos. Sistemas de bases de datos: diseño, implementación y administración. Quinta edición. México: International Thomson Editores, 2004. 838 p.

SENN, James. Análisis y Diseño de Sistemas de Información. Segunda Edición. México: Mc Graw-Hill, 1992. 242 p.

SILBERSCHATZ, Abraham; KORTH, Henry F. y SUDARSHAN. Fundamentos de bases de datos. Cuarta edición. Madrid: McGraw-Hill, 2002. 119 p.