



**UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO**

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

**ANÁLISIS Y DISEÑO DE UNA BASE DE DATOS PARA LA
MITIGACIÓN DE RIESGOS DERIVADOS DE LAS
INTERACCIONES MEDICAMENTOSAS EN HOSPITALES**

DISEÑO DE UN SISTEMA O PROYECTO

GERARDO CASTRO AGUILAR



MÉXICO, D.F.

2014



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



**UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO**

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

**ANÁLISIS Y DISEÑO DE UNA BASE DE DATOS PARA LA
MITIGACIÓN DE RIESGOS DERIVADOS DE LAS
INTERACCIONES MEDICAMENTOSAS EN HOSPITALES**

DISEÑO DE UN SISTEMA O PROYECTO

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMATICA

PRESENTA;

GERARDO CASTRO AGUILAR

ASESORA:

DRA. NADIMA SIMON DOMINGUEZ



MÉXICO, D.F.

2014

Dedicatorias

Quiero dar gracias a mis padres, Martha Aguilar Flores y Gerardo Castro Cruz simplemente porque sin ellos no sería quien soy, por los valores que me enseñaron, el sacrificio y cariño, los amo mucho.

A mis abuelos en especial a Silvano Castro Heras e Inés Flores Cortes, por siempre preguntar cómo iba mi proyecto, por apoyarme y creer en mí.

A mi hermana Alejandra Castro, amigos y compañeros por la ayuda que sin cuestionar me brindaron.

A mi familia, en especial a mi tío Chucho por su apoyo incondicional, consejos y compartir una parte de su tiempo conmigo y por animarme a poner este proyecto en marcha.

A Elizabeth Ortiz Rivera y a su familia por la paciencia, apoyo, consejos, ayuda, ideas, por siempre animarme y confiar en mí, muchas gracias por estar a mi lado Ely, te amo.

A todos ustedes por ser base para mi formación y aportar tantas cosas a i vita, les agradezco todo.

Agradecimientos

Al culminar este proyecto y sabiendo de antemano que con estas palabras no puedo expresar lo suficiente, quiero agradecer a todas aquellas personas que han colaborado conmigo ya que sin su ayuda no hubiera sido posible este logro que significa tanto para mí.

A la UNAM y a la Facultad de Contaduría y Administración, que han hecho de mí una persona preparada para el éxito, no puedo expresar lo orgulloso que me siento de pertenecer a esta Universidad.

A todos los profesores de la UNAM, quienes desde la preparatoria hasta la universidad ayudaron con mi formación académica, gracias por sus consejos y por compartir sus conocimientos y experiencia.

Al Dr. Jesús Ignacio Simón Domínguez, por haber confiado en mí y brindarme el apoyo y el ánimo que necesitaba. Gracias también por proporcionarme los datos necesarios, validar la información médica y por los comentarios y correcciones tan oportunas.

A mi tutora de tesis la Dra. Nadima Simón, por el tiempo dedicado en la orientación, seguimiento y revisión de esta tesis, gracias por ser una guía en este proyecto.

Al Dr. Raúl Ojeda, por tomarse el tiempo de leer este trabajo y darme su opinión, siendo él quien me abriera el camino cuando ya no encontraba hacia donde dirigir mi proyecto.

A todos ellos mi más grande agradecimiento.

Índice:

	Páginas
Introducción	1
Justificación del proyecto	2
Problemática	2
Objetivos generales y específicos	2
Método	3
Capítulo 1. Bases de datos	4
1.1 Conceptos Generales	6
1.2 Historia de las bases de datos	7
1.3 Arquitectura de los SGBD	12
1.4 Modelos de base de datos	16
1.5 Lenguajes	18
Capítulo 2. El modelo relacional	21
2.1 Introducción	21
2.2 Operaciones del modelo relacional	25
2.3 Reglas de integridad	25

Capítulo 3. Structured Query Language (SQL)	27
3.1 Introducción	27
3.2 Sentencias	28
3.2.1 Sentencias de definición	28
3.2.2 Sentencias de manipulación	32
3.2.3 Sentencias de control	36
Capítulo 4. Diseño de Base de Datos	39
4.1 Etapas del diseño de base de datos	39
4.2 Modelo Entidad Relación	40
4.2.1 Entidad	41
4.2.2 Atributos	42
4.2.3 Relaciones	43
4.2.4 Conectividad y Cardinalidad	43
4.2.5 Dependencias	44
Capítulo 5. MySQL	45
5.1 Características de MySQL	45
5.2 Interfaces de MySQL	45
5.3 Conexión a una base de datos.	46
5.4 Clientes y servidores	46

5.5 Tipos de datos	47
Capítulo 6. Interacciones Medicamentosas.....	51
6.1 Introducción	51
6.2 Importancia	52
6.3 Tipos de interacciones.....	52
6.4 Interacciones con alimentos	52
Capítulo 7. Caso Práctico	54
7.1 Creación de la Base de Datos	53
7.2 Pruebas y validacion.....	69
Conclusiones	72
Bibliografía.....	74

Introducción

Este trabajo fue realizado en el marco del macroproyecto de la Facultad de Contaduría y Administración titulado “Administración y sustentabilidad” coordinado por la Dra. Nadima Simón Domínguez. El diseño del sistema de riesgos derivados por las interacciones medicamentosas en hospitales, objetivo del presente trabajo, se elaboró en el área de Administración de Riesgos y Seguridad Hospitalaria de dicho macroproyecto, cuyo responsable es el Dr. Jesús Ignacio Simón Domínguez.

El uso de medicamentos constituye en la actualidad una de las principales causas de mortalidad en el mundo desarrollado, los cuales históricamente se han utilizado para salvar vidas y prevenir enfermedades. Sin embargo su utilización inadecuada los está convirtiendo en un importante problema de salud pública. (García et al, 2014). Atender esta problemática es fundamental para el desarrollo sustentable, entendido como “aquel que busca satisfacer las necesidades del presente sin comprometer la capacidad de las generaciones futuras de satisfacer sus propias necesidades” (UN, 1987: 41).

Los pacientes en estado crítico frecuentemente reciben múltiples medicamentos, los cuales además de ayudar en el tratamiento del paciente pueden producir reacciones que afecten su salud debido al alto riesgo de presentar interacciones medicamentosas. La relevancia clínica de las interacciones medicamentosas depende de varios factores, tales como las enfermedades específicas, la edad, el género, hábitos y factores genéticos.

La tecnología ha beneficiado enormemente diversos aspectos de la vida diaria en especial el manejo de la información; haciéndola más oportuna, precisa y veraz, reduciendo el tiempo de espera para muchos servicios. Los médicos cuentan con herramientas que lo orientan en la toma de decisiones, las cuales le proporcionan información útil para el manejo de los medicamentos: dosis, presentaciones, vía de administración, indicaciones, efectos secundarios y reacciones adversas

producidas por las interacciones.

El presente trabajo tiene por objetivo describir las etapas en las que se llevó a cabo este proyecto, así como subrayar la importancia de una base de datos de interacciones medicamentosas.

Justificación del proyecto

La cantidad de información que se genera con los nuevos medicamentos y los ya existentes es muy grande, por lo que se propone la creación de una base de datos que mejore la administración y procesamientos de datos dentro de los hospitales, con la finalidad de reducir el riesgo que existe en la prescripción de varios medicamentos.

Problemática

Un problema relacionado con medicamentos es un riesgo presente en cualquier paciente debido a interacciones y reacciones adversas a éstos, evidenciando la necesidad de crear una base de datos con el fin de minimizar los riesgos.

Objetivos generales y específicos

Objetivo general

- Creación de una base de datos que ayude en la correcta toma de decisiones de médicos y enfermeras. Dicha base de datos predecirá efectos secundarios que pudieran surgir del uso

de medicamentos y sus interacciones y además alertará sobre combinaciones que deberán ser descartadas para evitar serias consecuencias.

Objetivos específicos

- Facilitar la prescripción de medicamentos.
- Minimizar efectos secundarios.
- Evitar complicaciones por el uso de medicamentos.
- Destacar la importancia de las interacciones entre fármacos.
- Describir los tipos de interacciones entre medicamentos y alimentos.

Método

El sistema de gestión de base de datos seleccionado para la creación de la base de datos es MySQL. Para la creación de la base de datos se realizó una revisión bibliográfica exhaustiva en bases de datos, se realizó el análisis del problema y luego se recopiló la información. Después se seleccionó el manejador de base de datos MySQL para realizar el diseño de la base de datos y la carga de los datos. Una vez terminado el diseño se realizaron las pruebas y validación así como la documentación de la base de datos.

La fuente para extraer los datos requeridos fue proporcionada por el Dr. Jesús Ignacio Simón Domínguez, consultor médico de Tecnología Informática Moderna S.A. (TIMSA); dicha fuente consta de 25,000 medicamentos y sus interacciones, los datos se encontraban en forma descriptiva en texto libre.

Actualmente salen a la venta nuevos medicamentos con distintos principios activos, y es imposible para un médico memorizar tantos medicamentos y sus posibles reacciones.

Las reacciones negativas por un medicamento es un problema de tal magnitud, que causa más defunciones que los accidentes de automóvil, el cáncer de mama o el síndrome de inmunodeficiencia adquirida (SIDA). (Mora et al, 2006)

Gestión de riesgo

La administración de riesgos médicos y hospitalarios comparte elementos con la administración de riesgos de cualquier empresa, solamente que además de preservar el patrimonio de la institución también se preserva la salud y vida de los pacientes.

Los programas de Administración de Riesgos de los hospitales tradicionalmente se han concentrado en mejorar la seguridad de los equipos, de los empleados y de los pacientes, lo cual es importante y puede de hecho prevenir accidentes por los que el hospital podría resultar responsable.

Algo menos desarrollado en los hospitales es la administración de riesgos clínicos que consiste en la identificación temprana de aquellos eventos relacionados con el manejo del paciente que potencialmente podrían resultar en responsabilidad para el hospital o sus profesionales.

La Universidad de Harvard publicó en la década de los ochenta que el 4% de los pacientes hospitalizados en el estado de Nueva York en 1984 sufrieron eventos adversos, definidos éstos como daños ocasionados por el tratamiento médico. Si esta estadística se aplica en todos los Estados Unidos se calcula que más de un millón de pacientes sufren daños ocasionados por distintos tratamientos (aun sin mediar negligencia), y aproximadamente más de 180.000 personas mueren anualmente a causa de estos daños, superando ampliamente la mortalidad producida por los

accidentes automovilísticos (45.000). En el mismo estudio, los médicos revisores concluyen que el 60% de estos eventos se debieron, al menos en parte, a errores en el manejo del paciente, los cuales pudieron haber sido prevenidos. (Leape et al, 1991).

Capítulo 1. Bases de datos

1.1 Conceptos Generales

Para comprender claramente el concepto de base de datos es necesario entender primero la diferencia entre datos e información.

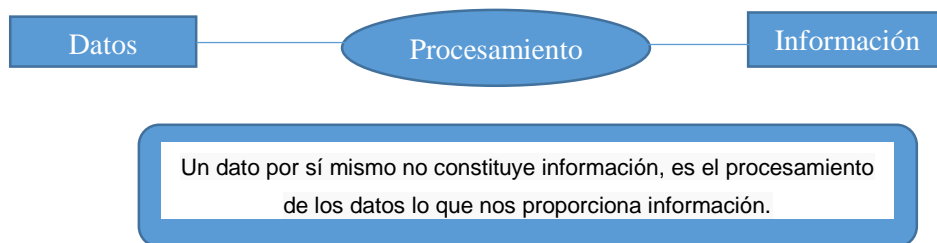
Dato: un dato es un conjunto de símbolos (numéricos, alfabéticos, algorítmicos, etc.) de un atributo o característica de una entidad; son hechos en bruto que aún no se han procesado para mostrar algún significado.

Información: es un conjunto de datos organizados ya procesados que tienen un significado, la información adecuada, pertinente y oportuna es la clave para una buena toma de decisiones.

Base de datos: se podrá definir una base de datos como un conjunto, colección o depósito de datos almacenados en un soporte informático que pueden organizarse e interrelacionarse de acuerdo con un modelo predefinido.

Figura 1.1

Definición de dato



Fuente: elaboración propia

Sistema de Gestión de Base de Datos (SGDB)

Un sistema de gestión de base de datos (SGDB) es un software o conjunto de programas que permite crear y mantener una base de datos. El SGDB actúa como interfaz entre los programas de aplicación (usuarios) y el sistema operativo. El objetivo general de un SGDB es proporcionar un entorno eficiente a la hora de almacenar y recuperar la información de la base de datos. (Costal Costa, 2009:7).

1.2 Historia de las bases de datos

Sistemas centralizados

En los años setenta los programas se ejecutaban por lotes (*batch*), este procesamiento se utilizaba para tareas repetitivas; se creaban archivos nuevos con datos duplicados (redundancia) para que los programas no leyeran demasiados archivos y el procesamiento fuera más rápido.

Con el paso del tiempo los programas permitían a los usuarios ver y actualizar los archivos on-line y mientras más se integraban las aplicaciones y se interrelacionaban los archivos fue necesario eliminar la redundancia.

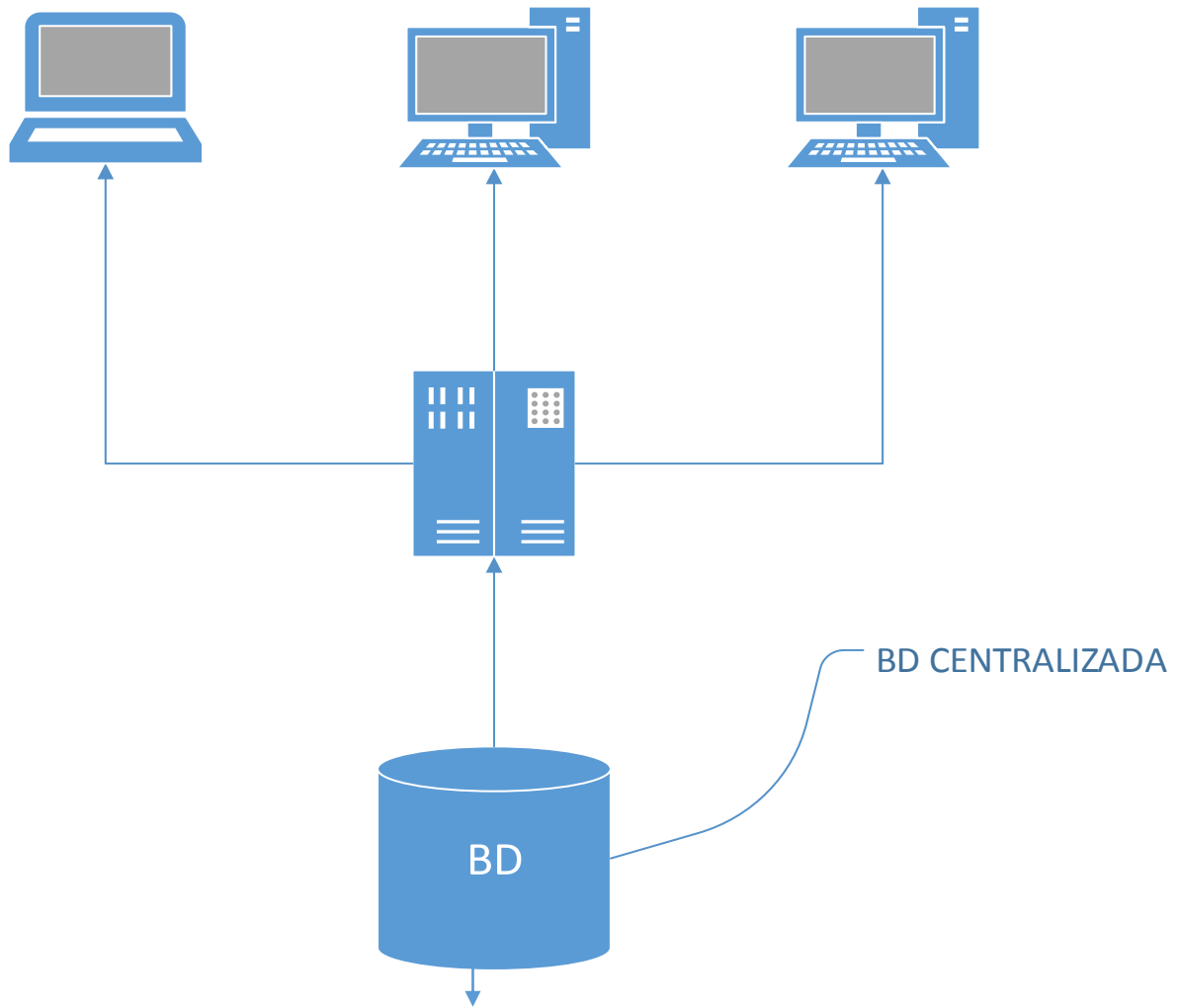
“Los conjuntos de archivos interrelacionados, con estructuras complejas y que compartían varios procesos recibieron el nombre de Data Banks y después, a inicios de los años setenta en Data Bases.” (Camps, 2009:7).

Durante la segunda mitad de los años setenta fueron apareciendo tipos de software más complicados, llamados Sistemas de Gestión de base de datos (SGDB). Los SGBD eran sistemas

totalmente centralizados, igual que los sistemas operativos de aquellos años.

Figura 1.2

Base de datos centralizada



Fuente: (Camps, 2010: 11).

SGBD Relacionales

Durante la década de los setenta aparecieron las primeras bases de datos relacionales (*Ingres Database, System R*), pero fue durante los años ochenta que se volvieron populares. Esta expansión fue gracias a las microcomputadoras que estaban al alcance para todas las empresas e instituciones, haciendo que el desarrollo de aplicaciones fuera más fácil.

Todo eso logró la expansión del uso de SGBD así como la estandarización dada por el ANSI (Instituto Nacional Estadounidense de Estándares), en el año de 1986, apareciendo la primera versión estándar del lenguaje *Structured Query Language* (SQL), el “SQL-86” o “SQL1”. (Date, 2001)

Base de datos distribuidas, C/S y 4GL

A principios de la época de los noventa dentro de las empresas no sólo había una computadora para toda la empresa si no que ya había computadoras por departamentos y personales; lo que creó la necesidad de tener una visión global de todas las relaciones que existían en las diferentes bases de datos. En atención a esta necesidad, surgieron los sistemas de gestión de base de datos actuales los cuales permiten que un programa pueda trabajar con diferentes bases de datos y a esto se le llamó como base de datos distribuidas.

Bases de datos distribuidas: “El soporte completo para las bases de datos distribuidas implica que una sola aplicación debe ser capaz de operar de manera transparente sobre los datos que están dispersos en una variedad de bases de datos diferentes, administradas por una variedad de distintos SGBD, ejecutadas en diferentes máquinas, manejadas por varios sistemas operativos diferentes y conectadas a redes de comunicación distinta; donde el término de manera transparente significa que la aplicación opera desde un punto de vista lógico como si todos los datos fueran manejados por un solo SGBD y ejecutados en una sola máquina” (Date, 2001: 672).

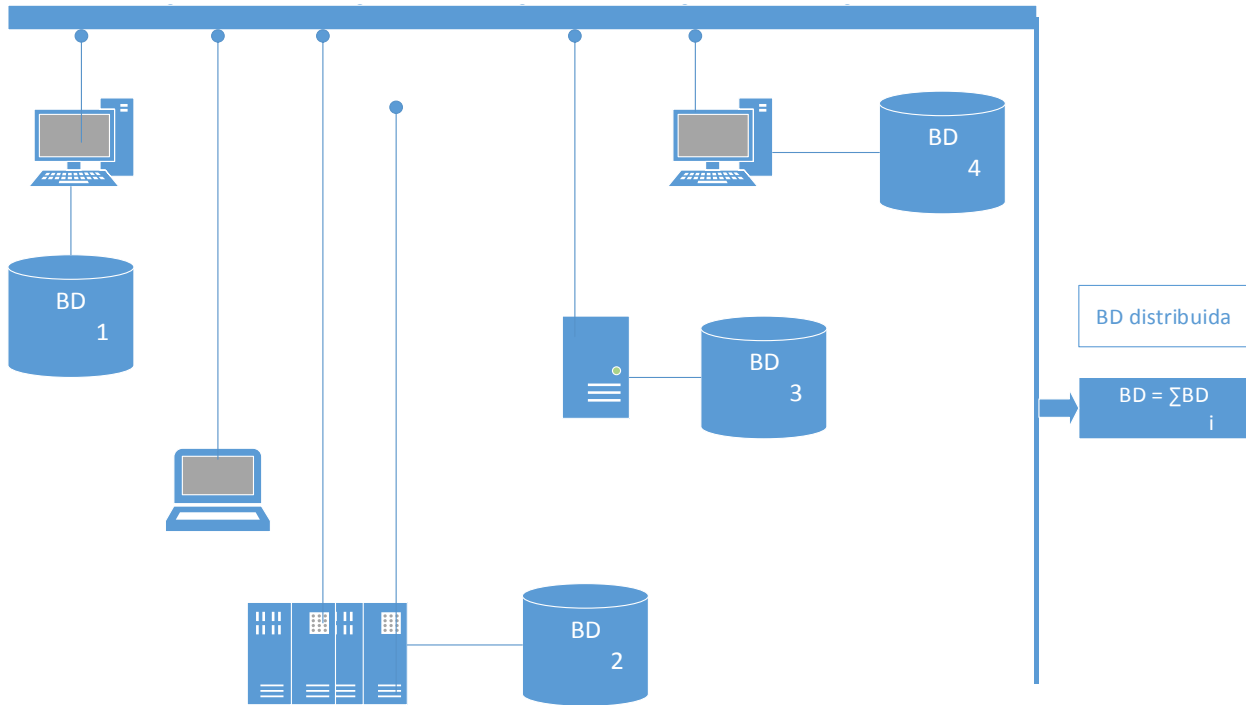
Un sistema de base de datos distribuida se da cuando existen varias bases de datos conectadas en diferentes redes trabajando como si fueran sólo una base de datos. Por la estandarización de SQL, los sistemas de gestión de base de datos de diferentes desarrolladores pueden trabajar sin ningún problema entre sí.

Las ventajas de las bases de datos distribuidas son, según Camps (2010), son:

- Disponibilidad, si un servidor cae, los demás seguirán disponibles, aunque también un inconveniente es que sólo los datos de los demás servidores estarán disponibles.
- La estructura misma de la base de datos refleja la de la empresa, donde la empresa está separada por departamentos y sus respectivos datos igual.
- Independencia de hardware.
- Independencia de SGBD.
- Opera continuamente.
- No depende de un sitio central.
- Disponibilidad.
- Independencia de ubicación.

Figura 1.3

Base de datos distribuidas



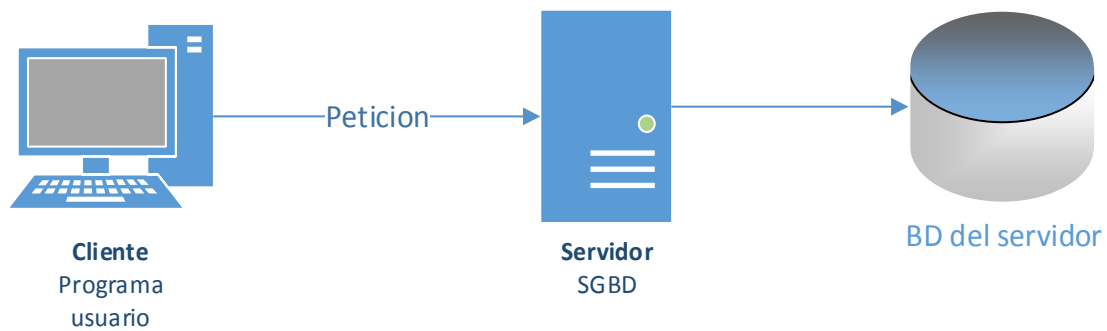
Fuente: (Camps, 2010: 11)

Generalmente para distribuir datos se utiliza el entorno cliente/servidor (C/S). Los SGDB del mercado están adaptados para este tipo de entorno. La arquitectura cliente/servidor permite que los usuarios trabajen en computadoras llamadas sistemas frontales (*front-end*) que interaccionan con servidores llamados posteriores (*back-end*), que proporcionan servicios como el acceso a una base de datos, la gestión de red y el almacenamiento de archivos. Una red de computadoras ofrece la plataforma de comunicación en la que numerosos clientes pueden interactuar con uno o más servidores. Esta interacción entre la aplicación que ejecutan los usuarios en sus sistemas frontales y el programa (generalmente una base de datos o un sistema operativo de red) en el servidor posterior se denomina relación cliente-servidor. Esto implica que el usuario dispone de una computadora con su propia capacidad de procesamiento, que ejecuta un programa que puede efectuar la interacción con el usuario y la presentación de la información.

Igualmente el éxito de los lenguajes de cuarta generación (4GL) permite modificar y consultar datos en los entornos C/S.

Figura 1.4

Cliente/Servidor



Fuente: (Camps, 2010: 11)

Las bases de datos han evolucionado para estar integradas a la multimedia, la orientación a objetos (OO), el internet y la web. El éxito de la programación orientada a objetos, ha hecho que se extienda a muchos campos incluso a las bases de datos.

1.3 Arquitectura de los SGBD

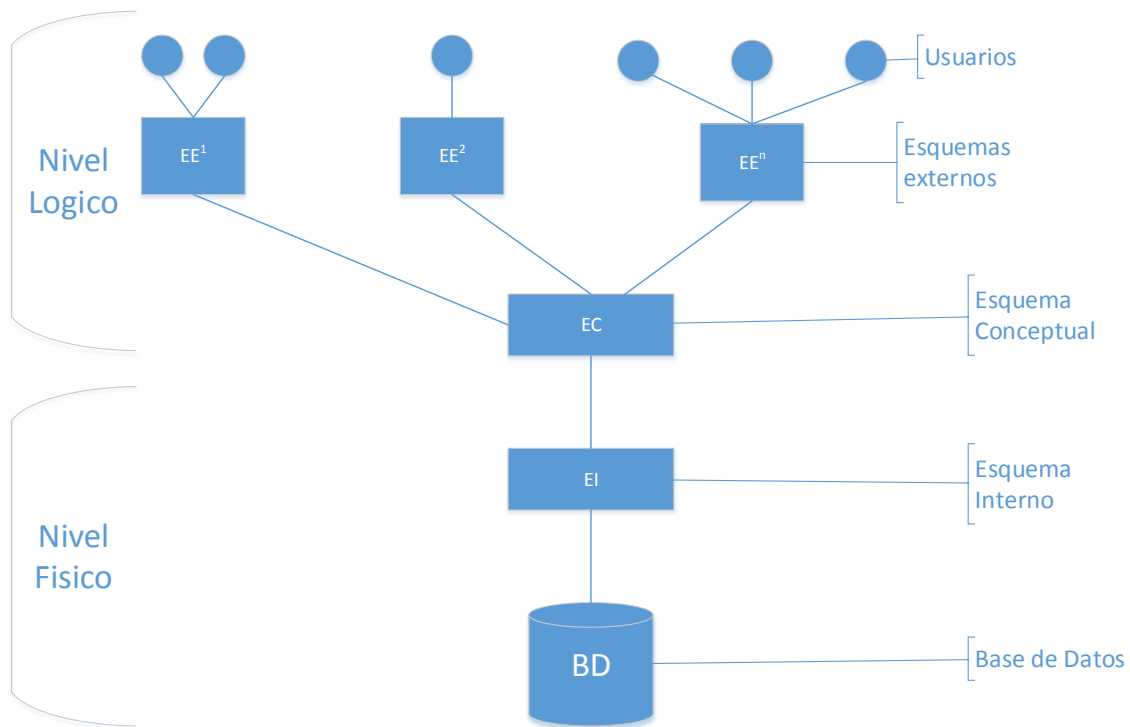
Esquemas y niveles

El ANSI (*American National Standards Institute*) es una organización sin fines de lucro, cuya función es supervisar el desarrollo de estándares en los Estados Unidos y es miembro de la Organización Internacional para la Estandarización (ISO); el ANSI durante el periodo de 1975-1982 desarrolló la arquitectura ANSI/SPARC, la cual recomendó que el SGBD tuviera tres niveles:

- Nivel externo: define los procesos usuarios, que denominamos esquemas externos.
- Nivel conceptual: que denominamos esquema conceptual y define cómo se organiza la información dentro de la base de datos, sirve de referencia para el resto de los esquemas.
- Nivel físico o interno: define las visiones lógicas del usuario.

Figura 1.5

Esquemas y niveles



Fuente: (Camps, 2010: 23)

- El esquema externo, se describe sólo la parte de la base de datos que interesa a un grupo de usuarios determinado, y oculta a ese grupo el resto de la base de datos.

- El esquema conceptual, se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones.
- El esquema interno, describe los detalles para el almacenamiento de la base de datos, como los métodos de acceso.

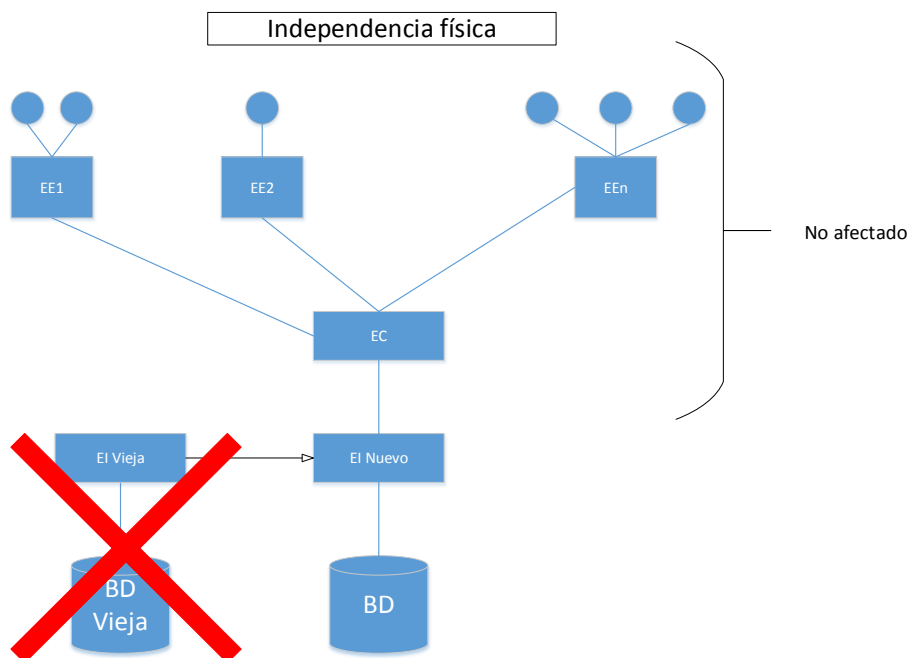
Independencia de los datos

La arquitectura de los tres niveles, nos proporciona dos tipos de independencia de los datos: la física y la lógica.

- Independencia física: cuando los cambios de la BD no afectan al mundo exterior (programas, usuarios directos).

Figura 1.6

Independencia física

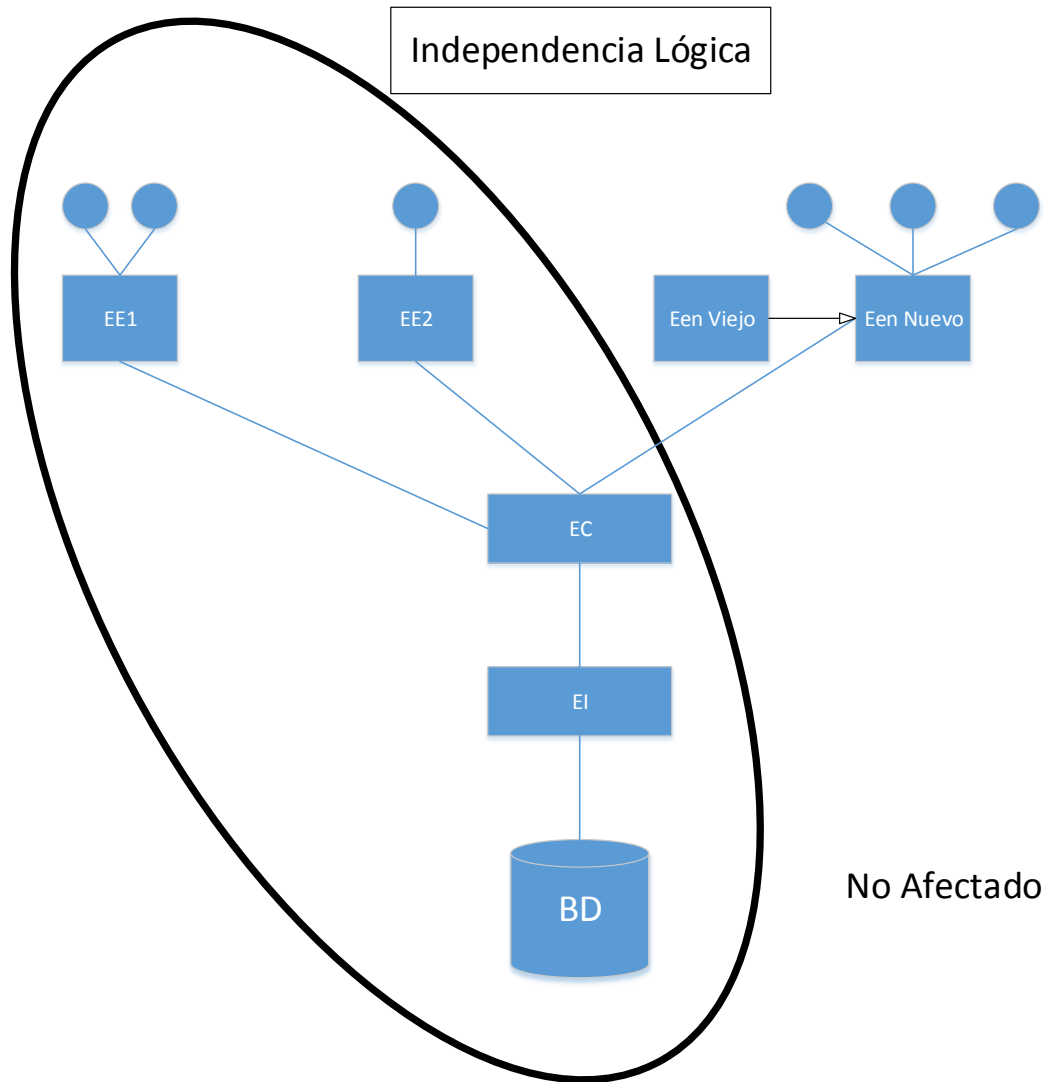


Fuente: (Camps, 2010: 25)

- Independencia lógica: Cuando los usuarios no se ven afectados por los cambios en el nivel lógico.

Figura 1.7

Independencia lógica



Fuente: (Camps, 2010: 26)

1.4 Modelos de base de datos

Las bases de datos representan una parte de la realidad, aunque el componente para el modelado de una base de datos relacional son las tablas, se utilizan otros componentes.

- Estructuras de datos: son las que pueden crear la BD (árboles, tablas, etc.)
- Restricciones de integridad: son las que el SGBD tiene que cumplir como los dominios, claves, etc.
- Operaciones: por ejemplo, en el modelo relacional es la operación SELECT, que es para seleccionar las filas.

Base de datos jerárquicos

La información es almacenada en una estructura jerárquica, organizando y enlazando los registros en estructura de árbol, cada elemento se denomina nodo, cada nodo tiene dependencia con otros nodos. Un nodo padre puede tener varios nodos hijo. Esta dependencia es de uno a muchos (1:M) un nodo hijo no puede tener más de un padre pero un padre sí puede tener varios hijos (Date, 2011).

Bases de datos de red

Este modelo es parecido al modelo jerárquico, su diferencia es que un mismo nodo tenga varios padres. (Date, 2011)

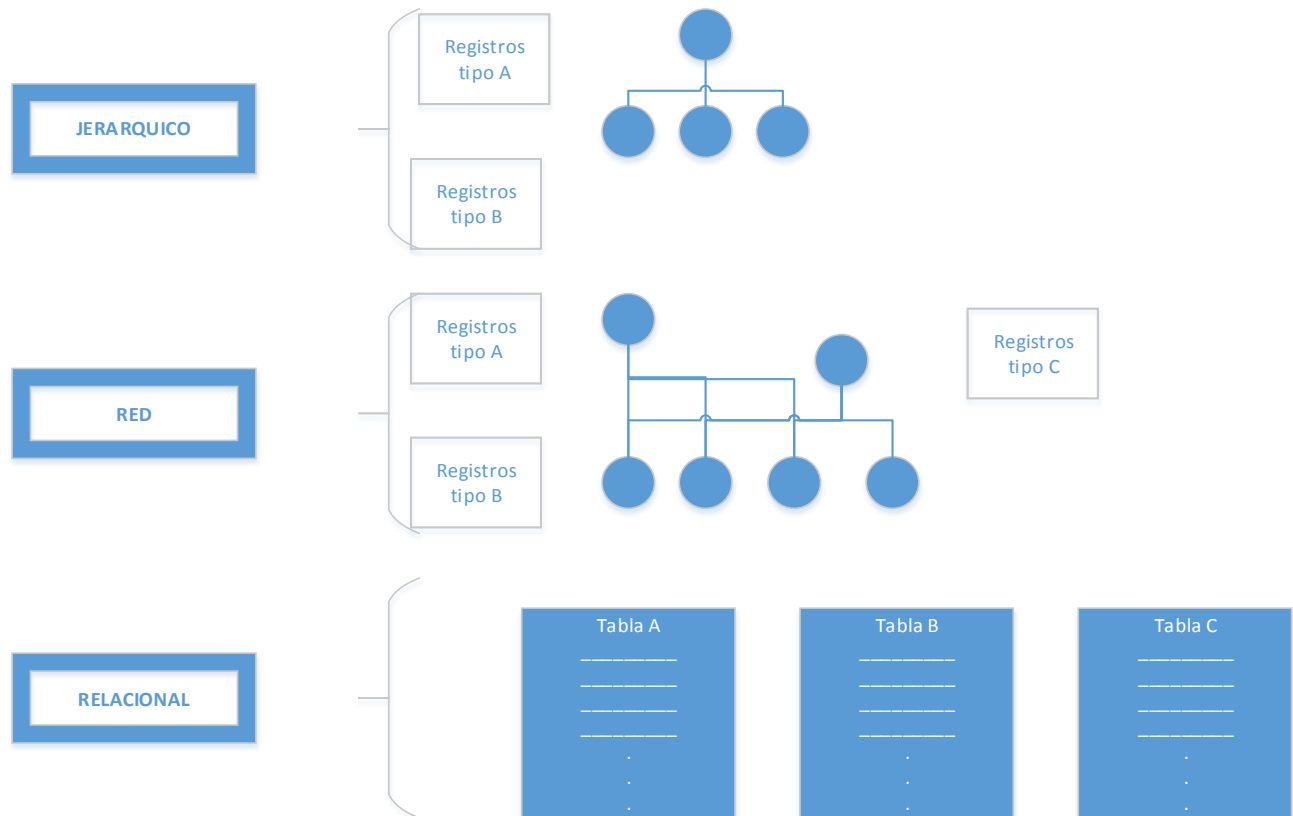
Bases de datos relacionales

Este tipo de base de datos es la más utilizada, la información se compone por tablas compuestas por registros (filas) que representan tuplas, y campos (columnas). Utilizan el lenguaje SQL. (Ramez et al, 2007).

Bases de datos orientadas a objetos

En una base de datos orientada a objetos, la información se representa mediante objetos como los que utiliza la programación orientada a objetos. Cuando se integran las características de una base de datos con las de un lenguaje de programación orientado a objetos, el resultado es un sistema gestor de base de datos orientada a objetos (*Object Database Management System*, por sus siglas en inglés, ODBMS). Un ODBMS incorpora muchos de los paradigmas de la programación orientada a objetos, como los tipos de datos abstractos, la encapsulación de operaciones y la herencia. Las bases de datos orientadas a objetos se diseñan para trabajar bien en conjunción con lenguajes de programación orientados a objetos como Java, C#, Visual Basic.NET y C++. Los ODBMS usan exactamente el mismo modelo que estos lenguajes de programación. (Ramez et al, 2007).

Modelos de bases de datos



Fuente: (Camps, 2010: 30)

1.5 Lenguajes

Existen muchos lenguajes diferentes, cada uno está pensado para diferentes usuarios:

- Usuarios expertos con conocimientos en informática, se necesitan lenguajes complejos para escribir procesos complejos.
- Usuarios finales, que solo harán consultas. Necesitan un lenguaje muy sencillo.

- Usuarios finales no informáticos, necesitarán lenguajes muy eficientes.

El Lenguaje SQL

El lenguaje SQL es el más utilizado, tiene tres tipos de instrucciones diferentes:

1. Tipo **DML (lenguaje de manipulación de datos)**: Ejemplo **SELECT, INSERT, UPDATE** y **DELETE** para hacer consultas y mantenimiento de los datos.
2. Tipo **DDL (lenguaje de definición de datos)**: Ejemplo: **CREATE TABLE** para crear tablas, columnas y restricciones.
3. Tipo control de entorno: ejemplo **COMMIT** y **ROLLBACK** para delimitar transacciones.

Respecto a **DML** hay dos tipos de lenguajes:

1. Los Lenguajes declarativos
2. Los Lenguajes procedimentales

La mayoría de los SGBD tienen SQL como lenguaje, aun así también ofrecen 4GL y herramientas visuales.

Lenguajes 4GL

Son de muy alto nivel, facilitan el tratamiento de la BD. Traducen lo que hace el usuario a instrucciones SQL, esta traducción se hace mediante el intérprete de SQL integrado en el SGBD.

Herramientas o interfaz visual

Se emplea una interfaz muy fácil de manejar, la cual utiliza la base de datos al estilo de ventanas e íconos como en el sistema operativo Windows; traduce lo que hace el usuario a instrucciones SQL mediante la compilación.

Al escribir un programa que trabaje con la base de datos utilizando algún lenguaje de programación (C, Pascal, Cobol, PL/I, Basic, JAVA, etc.); se pueden realizar llamadas a funciones o lenguaje hospedado.

Llamada a funciones: existen funciones especializadas en BD, por ejemplo las librerías ODBC, las funciones se encargarán en enviar las instrucciones al SGBD.

Lenguaje hospedado: es utilizar un pre-compilador que incluye instrucciones en el lenguaje de la base de datos, haciendo que el lenguaje de programación sea el lenguaje anfitrión y el lenguaje de la base de datos sea el hospedado.

Para lo descrito en esta sección se consultó el texto *Aprende SQL*.

Capítulo 2. El modelo relacional

2.1 Introducción

El software de procesamiento de datos de más uso en la actualidad son los sistemas de gestión de base de datos relacionales (SGBDR); éstos representan la segunda generación de SGBD y están basados en el modelo de datos relacional propuesto por E.F. Codd (1985a).

Los principios del modelo de datos relacionados fueron dados por E.F. Codd en los años 1969 y 1970, creando las 12 reglas de Codd (1985b).

Las 12 reglas de Codd para una base de datos relacional

Regla 1. Información.

Toda la información de una base de datos relacional debe estar representada lógicamente como valores en columnas y en renglones dentro de tablas.

Regla 2. Acceso garantizado.

Todo valor en una tabla está garantizado para ser accesible mediante una combinación de nombre de tabla, valor de llave primaria y nombre de columna.

Regla 3. Tratamiento sistemático de nulos.

Los nulos deben ser representados y tratados de forma sistemática, independiente del tipo de datos.

Regla 4. Catálogo dinámico en línea basado en el modelo relacional.

Los metadatos, deben guardarse y manejarse como datos ordinarios, es decir, en tablas dentro de la base de datos. Esos datos deben estar disponibles a usuarios autorizados que usen el lenguaje relacional estándar de base de datos.

Regla 5. Sublenguaje completo de datos.

La base de datos relacional puede soportar muchos lenguajes, pero debe soportar un lenguaje bien definido y declarativo, con soporte para definición de datos, definición de vista, manipulación de datos, restricciones de integridad, autorización y administración de transacción.

Regla 6. Actualización de vista

Cualquier vista que sea teóricamente actualizable debe ser actualizable a través del sistema.

Regla 7. Inserción, actualización y eliminación de alto nivel.

La base de datos debe soportar inserciones, actualizaciones y eliminaciones a nivel del conjunto.

Regla 8. Independencia física de datos.

Los programas de aplicación y dispositivos *ad hoc* no son afectados lógicamente cuando se cambian métodos de acceso físicos y estructuras de almacenamiento.

Regla 9. Independencia lógica de datos.

Los programas de aplicación y dispositivos *ad hoc* no son afectados lógicamente cuando se hacen cambios a las estructuras de una tabla que preserven los valores originales de tablas.

Regla 10. Independencia de integridad.

Todas las restricciones de integridad relacionales deben ser definibles en el lenguaje relacional y guardado en el catálogo del sistema, no al nivel de aplicación.

Regla 11. Independencia de distribución.

Los usuarios finales y programas de aplicación no están enterados y no son afectados por la ubicación de datos.

Regla 12. No subversión.

Si el sistema soporta un acceso de bajo nivel a los datos, no debe haber forma de saltarse las reglas de integridad de la base de datos.

Regla cero.

Todas las reglas precedentes están basadas en la noción de que para que una base de datos sea considerada relacional, debe usar sus dispositivos relacionales exclusivamente para manejar la base de datos.

Terminología

El modelo relacional está basado en el concepto matemático de una relación, los datos están estructurados mediante relaciones (tablas), las cuales tienen un nombre y están compuestas por atributos (columnas). Cada tupla (fila) contiene un valor por cada atributo.

- Relación: es una tabla con columnas y filas.
- Atributo: es una columna nominada de una relación.
- Dominio: conjunto de valores permitidos para uno o más atributos.

- Tupla: es una fila de una relación.
- Grado: el grado de una relación es el número de atributos que contiene.
- Cardinalidad: la cardinalidad de una relación es el número de tuplas que contienen.

Tabla 2.1

Nombres del modelo relacional

Nombre formal	Nombre informal
Relación	Tabla
Atributo	Columna o campo
Grado	No de Columnas
Tupla	Fila o registro
Cardinalidad	No de filas
Clave Primaria	Identificador único
Dominio	Conjunto de valores legales

Fuente: (Nevado, 2010: 61)

El objetivo del modelo de datos relacional es que la base de datos sea vista y percibida como una estructura lógica, simple y uniforme. El modelo relacional toma en cuenta aspectos de los datos, como la estructura, la manipulación y la integridad.

Propiedades de las relaciones

Las relaciones tienen las siguientes características, según Nevado (2010):

- Cada relación tiene un nombre que es distinto a las demás relaciones.
- Los valores de los atributos son atómicos. En cada atributo toma un solo valor, se dice que las relaciones están normalizadas.
- Los atributos no se deben llamar igual.
- Los atributos no están ordenados.
- No existen tuplas duplicadas.
- Las tuplas no están ordenadas.

2.2 Operaciones del modelo relacional

Las operaciones del modelo relacional deben permitir manipular los datos, que incluye la actualización y la consulta.

Existen tres operaciones básicas de actualización, según Nevado (2010):

- A. Inserción, que sirve para añadir una o más tuplas a una relación.
- B. Borrado, elimina una o más tuplas de una relación.
- C. Modificación, para alterar los valores que tienen una o más tuplas de una relación para uno o más de sus atributos.

2.3 Reglas de integridad

Una base de datos contiene datos que reflejan la realidad. Para que los datos sean íntegros deben

cumplir varias condiciones, las cuales garantizan la integridad de los datos y pueden ser de dos tipos:

- 1) Restricciones de integridad de usuario: son condiciones específicas de una base de datos concreta; son las que se deben cumplir en una base de datos particular con unos usuarios concretos, pero que no son necesariamente relevantes en otra base de datos.
- 2) Reglas de integridad de modelo: son condiciones más generales propias de un modelo de datos y se deben cumplir en toda base de datos que siga dicho modelo. (Coronel, 2003).

Regla de integridad de unicidad de la clave primaria

La regla de integridad de unicidad de la clave primaria establece que toda clave primaria no debe tener valores repetidos.

Regla de integridad de entidad de la clave primaria

La regla de integridad de entidad de la clave primaria dispone que los atributos de la clave primaria de una relación no pueden tener valores nulos.

Esta regla es necesaria para que los valores de las claves primarias puedan identificar las tuplas individuales de las relaciones; si las claves primarias tuviesen valores nulos, es posible que algunas tuplas no se pudieran distinguir.

Regla de integridad referencial

Esta regla está relacionada con el concepto de clave foránea. Determina que todos los valores que toma una clave foránea deben ser valores nulos o valores que existen en la clave primaria a la que

hacen referencia. Si un valor de una clave foránea no estuviese presente en la clave primaria correspondiente, representaría una referencia o una conexión incorrecta.

Capítulo 3. *Structured Query Language (SQL)*

3.1 Introducción

SQL es un lenguaje de cuarta generación, diseñado para la definición, manipulación y control de las bases de datos relacionales. Es un lenguaje muy parecido al inglés y sumamente expresivo, actualmente es el más empleado en este tipo de base de datos.

SQL fue desarrollado por IBM a principios de los años setentas, implementado por System R. El proyecto de IBM dio como resultado el lenguaje SEQUEL (*Structured English Query Language*); en 1976 se hizo una versión revisada llamada SEQUEL/2 pero fue denominado SQL, ya que SEQUEL fue una marca registrada por una compañía de aviones llamada *Hawkey Siddeley*.

En 1982, *American National Standards Institute (ANSI)* convirtió a SQL en el lenguaje estándar de bases de datos relacionales y en 1978 se convirtió en el lenguaje estándar de ISO (*International Standards Organization*), también aceptado como estándar por FIPS (*Federal Information Processing Standard*), X/Open y SAA (*System Application Architecture*).

A lo largo del tiempo SQL ha sido revisado y ampliado muchas veces. La última revisión para SQL fue adoptada oficialmente en Diciembre del 2011 llamándose SQL: 2011.

SQL se convirtió en el lenguaje estándar de base de datos con mayor aceptación en todo el mundo; el otro lenguaje de base de datos en red (*Network Database Language, NDL*) basado en el modelo de red CODASYL, tiene muy poco éxito. La mayoría de los fabricantes de base de datos crean sus productos basados en SQL.

Ventajas de SQL:

- Es utilizado en la mayoría de los manejadores de base de datos actuales.
- Fácil de comprender.
- Al ser un lenguaje completo no es necesario combinar SQL con algún lenguaje anfitrión.

3.2 Sentencias

Para el desarrollo de esta sección, se consultó *MySQL documentation; reference manual*.

3.2.1 Sentencias de definición

Las sentencias de definición son las siguientes:

Sentencia CREATE

Este comando se utiliza para crear bases de datos, tablas, dominios, aserciones y vistas.

- Sintaxis de **CREATE DATABASE**

```
CREATE DATABASE Nombre_BD;
```

CREATE DATABASE: crea una base de datos con el nombre dado, ocurre un error si la base de datos existe y no especifica **IF NOT EXISTS**.

En MySQL 5.0, las opciones **create_specification** pueden darse para especificar característica de la base de datos.

CREATE SCHEMA es un sinónimo de **CREATE DATABASE** que puede usarse desde MySQL 5.0.2

- Sintaxis de **CREATE INDEX**

Se utiliza para crear índices en las tablas. Los índices son creados para encontrar datos más rápidamente, los usuarios no ven los índices. El actualizar una tabla con índices llevará más tiempo que una tabla sin índices, dado que el índice también necesita ser actualizado.

```
CREATE INDEX nombredelIndice  
ON Nombre_Tabla (Nombre_Columna);
```

- Sintaxis de **CREATE TABLE**

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_tabla  
[(create_definition,...)]  
[table_options][select_statement]
```

Crea una tabla con el nombre dado, ésta se crea en la base de datos actual, ocurre un error si la tabla existe, si no hay base de datos actual o si la base de datos no existe. MySQL 5.0 soporta la sentencia **IF NOT EXISTS** para que no ocurra un error si la tabla no existe.

Sentencia ALTER

Este comando se utiliza para modificar tablas.

- Sintaxis de **ALTER DATABASE**

```
ALTER {DATABASE | SCHEMA} [Nombre_BD]
alter_specification [, alter_specification]...
alter_specification;
[DEFAULT] CHARACTER SET charset_name
[DEFAULT] COLLATE collation_name
```

ALTER DATABASE permite cambiar las características globales de una base de datos. La cláusula **CHARACTER SET** cambia el conjunto de caracteres por defecto de la base de datos. La cláusula **COLLATE** cambia la colación por defecto de la base de datos.

- Sintaxis de **ALTER TABLE**

Para agregar una columna de una tabla:

```
ALTER TABLE Nombre_Tabla
ADD Nombre_columna datatype
```

Para eliminar una columna de una tabla:

```
ALTER TABLE Nombre_Tabla
DROP COLUMN Nombre_columna
```

ALTER TABLE permite cambiar la estructura de una tabla existente. Por ejemplo se puede añadir o borrar columnas, crear o destruir índices, cambiar tipo de columnas o renombrarlas a la misma tabla.

ALTER TABLE funciona creando una copia temporal de la tabla original. La alteración se realiza en la copia, luego la tabla original se borra y se renombra la nueva. Mientras se ejecuta **ALTER TABLE** la tabla original es legible por otros clientes. Las actualizaciones y escrituras en la tabla se esperan hasta que la nueva tabla esté lista, luego se redirigen automáticamente a la nueva tabla sin ninguna actualización fallida.

Sentencia DROP

Este comando se utiliza para borrar bases de datos, tablas, dominios, aserciones y vistas.

- Sintaxis de **DROP DATABASE**

```
DROP {DATABASE | SCHEMA} [IF EXIST] Nombre_BD
```

Borra todas las tablas en la base de datos y borra la base de datos, **IF EXISTS** se usa para evitar un error si la base de datos no existe.

DROP SCHEMA es un sinónimo de **DROP DATABASE** y se puede utilizar desde MySQL 5.0.2.

- Sintaxis de **DROP INDEX**

```
DROP INDEX Nombre_Index ON Nombre_tabla
```

Borra el índice llamado **Nombre_Index** de la tabla **Nombre_Tabla**.

- Sintaxis de **DROP TABLE**

```
DROP [TEMPORARY] TABLE [IF EXISTS]
Nombre_Tabla [, Nombre_Tabla]...
[RESTRIC I CASCADE]
```

Borra una o más tablas. Debe tener el permiso **DROP** para cada tabla. Todos los datos de la definición de tabla son *borrados*

Se utiliza IF EXISTS para evitar un error para tablas que no existan.

Sentencia RENAME

- Sintaxis de **RENAME TABLE**

```
RENAME TABLE Nombre_Tabla TO new_Nombre_Tabla
[,Tabla2_Nombre TO new_Tabla2_Nombre]...
```

Este comando renombra una o más tablas.

3.2.2 Sentencias de manipulación

Las sentencias de manipulación son las siguientes:

- **INSERT:** para insertar nuevos datos en una tabla.
- **UPDATE:** para actualizar los datos de una tabla.
- **DELETE:** para borrar los datos de una tabla.
- **SELECT :** para consultar los datos de la base de datos.

Sentencia INSERT

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] Tabla_Nombre [(Columna_Nombre,...)]
```

INSERT: inserta nuevos registros en una tabla existente. Las formas **INSERT ... VALUES** y las de **INSERT ... SET** del comando insertan registros basados en valores explícitamente especificados. La forma **INSERT ... SELECT** inserta registros seleccionados de otra tabla o tablas.

Sentencia UPDATE

- Sintaxis de **UPDATE**

```
UPDATE [LOW_PRIORITY] [IGNORE] Nombre_Tabla
SET columna1=valor1 [,columna2=valor2...]
[WHERE donde_definicion]
[ORDER BY...]
[LIMIT numero_filas]
```

El comando **UPDATE** actualiza columnas en registros de tabla existentes con nuevos valores. La cláusula **SET** indica qué columna modificar y los valores que puede recibir. La cláusula **WHERE**, si se da, especifica qué registros deben actualizarse. De otro modo, se actualizan todos los registros. Si la cláusula **ORDER BY** se especifica, los registros se actualizan en el orden que se especifica. La cláusula **LIMIT** es el límite de registros a actualizar.

Sentencia DELETE

- Sintaxis de **DELETE**

```
DELETE FROM Nombre_Tabla  
WHERE Alguna_columna=Algun_valor;
```

DELETE borra los registros de **Nombre_Tabla** que satisfacen la condición dada por **WHERE**, y retorna el número de registros borrados.

Si realiza un comando **DELETE** sin cláusula **WHERE** se borran todos los registros.

Sentencia SELECT

- Sintaxis de **SELECT**

La sentencia **SELECT** extrae y visualiza los datos de una o más tablas de la base de datos. Es una sentencia potente, equivale a las operaciones de Selección, Proyección y Combinación del algebra relacional en una sola sentencia.

SELECT también puede usarse para recuperar registros computados sin referencia a ninguna tabla.

Para hacer consultas sobre una tabla con el SQL se utiliza la sentencia **SELECT FROM**.

```
SELECT [DISTINCT / ALL] { * [ExpresionOColumnas[AS Nombrenuevo]] }  
  
FROM Nombre_Tablas [alias]  
  
WHERE condición  
  
GROUP BY listaColumnas [HAVING condición]  
  
ORDER BY listaColumnas
```

```
SELECT * FROM Nombre_Tabla
```

ExpresionOColumna representa un nombre de columna o una expresión.

Nombre_Tabla es el nombre de la tabla que ya existe en la base de datos y alias es una abreviatura opcional para Nombre_Tabla.

FROM: especifica la tabla o tablas que se usarán.

WHERE: extrae los resultados que cumplen una condición específica.

GROUPBY: los resultados de los datos solicitados se agrupan en una o más columnas.

HAVING: especifica una condición de búsqueda para un grupo.

SELECT: selecciona los datos de una base de datos.

ORDERBY: ordena los resultados en un conjunto de una o más columnas.

3.2.3 Sentencias de control

Además de definir y manipular la base de datos es importante establecer mecanismos de control para resolver problemas de concurrencia de usuarios y garantizar la seguridad de los datos.

Autorizaciones

Los privilegios sobre la base de datos los tiene el propietario, pero no es el único que accede a ésta, SQL ofrece sentencias para autorizar y desautorizar a otros usuarios.

1) Autorización

Para autorizar se dispone de la siguiente sentencia:

```
GRANT privilegios ON objeto TO usuarios  
[WITH GRANT OPTION];
```

Tenemos que:

- a) Los privilegios puede ser:
 - **ALL PRIVILEGES:** todos los privilegios sobre el objeto especificado.
 - **USAGE:** utilización del objeto especificado; en este caso el dominio.
 - **SELECT:** consultas.
 - **INSET [(columnas)]:** inserciones. Se puede concretar de qué columnas.

- **UPDATE** [(columnas)]: modificaciones. Se puede concretar de qué columnas.
- **DELETE**: borrados.
- **REFERENCES** [(columna)]: referencia del objeto en restricciones de integridad. Se puede concretar con columnas.

b) El objeto debe ser:

- **DOMAIN**: dominio
- **TABLE**: tabla
- **Vista**

c) Usuarios puede ser cualquiera: **PUBLIC**, o bien una lista de los identificadores de los usuarios que queremos autorizar.

d) La opción **WITH GRANT OPTION** permite que el usuario que autoricemos pueda, a su vez, autorizar a otros a acceder al objeto con los mismos privilegios con los que ha sido autorizado.

Desautorizaciones

Para desautorizar, se dispone de la siguiente sentencia:

```
REVOKE [GRANT OPTION FOR] privilegios ON  
objeto FROM usuarios [RESTRICT/CASCADE];
```

Tenemos que:

- a) Privilegios, objeto y usuarios son los mismos que para la sentencia **GRANT**.
- b) La opción **GRANT OPTION FOR** se utilizaría en el caso que se quiera quitar el derecho a autorizar (**WITH GRANT OPTION**).
- c) La opción **CASCADE** hace que se queden autorizados todos a la vez.
- d) La opción **RESTRICT** no nos permite desautorizar a unos usuarios si éste ha autorizado a otros.

Capítulo 4. Diseño de Base de Datos

Una parte importante de las bases de datos es su diseño, en el cual se deben tomar en cuenta las características requeridas por los datos para transformarlos en información que le sea útil al usuario, para ello el diseño de la base debe definir claramente la estructura de los datos. El diseño de la base debe construir el modelo de los datos, y el más utilizado actualmente es el Modelo Entidad-Relación.

4.1 Etapas del diseño de base de datos

Una base de datos bien diseñada debe permitir el acceso a la información exacta y actualizada, puesto que un diseño correcto es esencial para lograr los objetivos fijados para la base de datos.

Existen tres etapas principales para desarrollar un diseño apropiado de la base de datos: etapa del diseño conceptual, etapa del diseño lógico y etapa del diseño físico.

1) Etapa del diseño conceptual

En esta primera etapa se utiliza la información de los requerimientos de los usuarios para así obtener la estructura de la información, se trata de representar la estructura que tendrá la base de datos, sin importar qué tipo de lenguaje de programación se utilizará y qué base de datos (relacional, orientado a objetos, jerárquica, etc.); y está totalmente aislada del SGBD de destino.

2) Etapa del diseño lógico

Se basa en el resultado de la etapa del diseño conceptual, éste si se ajusta al tipo de base de datos que se utilizará (relacional, orientado a objetos, jerárquica, etc.) En el caso de las bases de datos relacionales en esta etapa se obtienen las relaciones con sus atributos, claves primarias y claves foráneas.

3) Etapa del diseño físico:

Última etapa del diseño, en la cual básicamente se establece cómo se implementará físicamente la base de datos. En esta etapa se producen las especificaciones reales, tanto para el hardware, el software, medios de entrada/salida, etc.; es decir, el diseño físico estará adaptado a un SGBD en específico.

Para una base de datos relacional en el diseño físico se debe:

- Crear las tablas y relaciones, además de las restricciones de las mismas.

- Establecer cómo se tendrá acceso a los datos.
- Modelar las medidas de seguridad del sistema.

El proceso de diseño de una base de datos se guía por algunos principios. El primero de ellos es que se debe evitar la información duplicada o, lo que es lo mismo, los datos redundantes, porque malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias. El segundo principio es que es importante que la información sea correcta y completa. Si la base de datos contiene información incorrecta, los informes que recogen información de la base de datos contendrán también información incorrecta y, por tanto, las decisiones que se tomen a partir de esos informes estarán mal fundamentadas.

4.2 Modelo Entidad Relación

El Modelo Entidad Relación (Modelo ER) propuesto por Peter Chen en 1970 presenta el modelo como una vista unificada de datos. Este modelo se centra en la estructura lógica y abstracta de los datos, como representación del mundo real, independientemente de las características físicas. (Nevado, 2010: 39).

El Modelo ER simplifica el diseño de la base de datos, se enfoca primeramente en las entidades y sus relaciones, posteriormente se enfoca en detallar la información mediante los atributos y restricciones que se pudieran aplicar.

Los elementos principales del Modelo ER son las entidades, atributos y sus relaciones.

4.2.1 Entidad

Las entidades son cualquier cosa o parte del mundo que son distinguibles del resto y que existen por sí solas. Es la cosa de la cual se desea obtener y almacenar información y en el modelo ER se representan gráficamente con rectángulo llevando por dentro el nombre de la entidad.

Tipos de Entidad

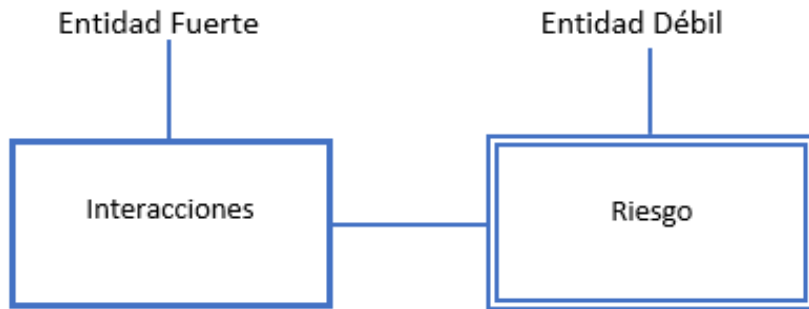
Las entidades se dividen en: fuertes y débiles.

Fuertes: las entidades fuertes no dependen de la existencia de otra entidad y poseen una clave principal en sus atributos. Se representan gráficamente con un rectángulo.

Débiles: este tipo de entidad sí depende de la existencia de otra entidad, si la entidad fuerte de la que depende desaparece, ésta inmediatamente desaparecerá también. No cuentan con clave principal en sus atributos y se representan gráficamente con dos rectángulos centrados.

Figura 4.1

Ejemplo de tipos de entidad



Fuente: elaboración propia

4.2.2 Atributos

Un atributo es una característica propia de la entidad, la cual deseamos almacenar ya que resulta interesante. Se representan gráficamente por círculos que cuelgan de las entidades o relaciones.

Simples

Aquellos atributos que no se pueden dividir y que cuentan sólo con un componente. Por ejemplo SEXO (Femenino, Masculino)

Compuestos

Son aquellos atributos que podemos dividir en más atributos. Por ejemplo NOMBRE, podemos dividirlo en APELLIDO PATERNO y APELLIDO MATERNO; es preferible para facilitar las consultas que se transformen los atributos compuestos en atributos simples.

Valor Sencillo

Atributo que únicamente puede tener un valor. Por ejemplo la FECHA DE NACIMIENTO, una persona sólo puede tener una sola fecha de nacimiento, un atributo monovaluado no necesariamente es un atributo simple; en este ejemplo podríamos dividir el atributo en año, mes y día.

Valor Múltiple

Atributos que pueden poseer muchos valores. Ejemplo, tenemos el atributo color para una casa, la cual puede tener más de un color para la puerta, ventanas, interiores, etc.

Derivados

Se llaman atributos derivados a aquellos que se obtienen a partir de otros y que no necesariamente se encuentran físicamente en la base de datos. Tenemos por ejemplo la EDAD de un empleado, la cual la podemos calcular a partir de la diferencia entre la fecha actual y la fecha de nacimiento de dicho empleado.

4.2.3 Relaciones

Una relación es una asociación entre dos a más entidades, no importando si son o no del mismo conjunto de entidades.

A las entidades involucradas en una relación se les denomina entidades participantes. El número de entidades participantes determinan el grado de la relación; esto es, si se involucran dos entidades, tenemos una relación binaria, si se involucran más de dos entidades se les llama relaciones n-arias.

4.2.4 Conectividad y cardinalidad

La cardinalidad de una relación refleja el número específico de ocurrencias de un tipo de entidad asociadas con el número de ocurrencias de la otra entidad relacionada. Se representa gráficamente a un lado de la entidad con el formato (x, y) . En el cual, el valor mínimo se representa con la x y el valor máximo con la y .

El termino conectividad hace referencia a la clasificación de las relaciones, esto es uno a uno (1:1), uno a muchos (1:M) y muchos a muchos (M:N).

Tenemos por ejemplo, una entidad Doctor y una entidad Paciente donde se pueden dar estas correspondencias de conectividad:

- Un doctor puede atender únicamente a un paciente (1:1),
- Un doctor puede atender a varios pacientes (1:M),
- Varios doctores pueden atender a distintos pacientes (M:N).

4.2.5 Dependencias

En algunos casos, una entidad individual sólo puede existir si hay como mínimo otra entidad individual asociada con ella mediante una relación binaria determinada. En estos casos, se dice que esta última entidad es una entidad obligatoria en la relación. Cuando esto no sucede, se dice que es una entidad opcional en la relación.

Para la elaboración de esta sección se consultó *SQL Tutorial*.

Capítulo 5. MySQL

5.1 Características de MySQL

MySQL es un gestor de base de datos. Se caracteriza por su facilidad de uso, sencillo de descargar e instalar, su libre distribución bajo los términos de la licencia GPL o uso comercial, su grado de estabilidad y rápido desarrollo; esto hace de MySQL uno de los gestores de bases de datos más populares del mercado.

MySQL está disponible para múltiples plataformas; las diferencias entre sistemas operativos es nulo, permitiendo trabajar sin ningún problema ya que permite utilizar la herramienta de mysql-client la cual trabaja e interactúa en un servidor instalado localmente o a través de internet.

MySQL es una opción muy popular de la base de datos para uso en aplicaciones WEB, utilizadas en LAMP (Ejemplo Apache, WAMP).

MySQL es popular por estas características:

- MySQL está desarrollado en C/C++.
- Se puede utilizar como cliente-servidor o incrustado en aplicaciones.
- Cuenta con muchos tipos de datos.
- Soporta múltiples métodos de almacenamiento de tablas.
- Su administración se basa en usuarios y privilegios.
- Estabilidad

5.2 Interfaz de MySQL

En MySQL los usuarios pueden utilizar la interfaz de línea de comandos o el uso del front-end MySQL Workbench que es desarrollado por Oracle; MySQL Workbench es una herramienta que integra de forma visual el desarrollo, administración, diseño, creación y mantenimiento de las bases de datos. También se puede administrar con aplicaciones web que crean y administran las bases de datos MySQL.

5.3 Conexión a una base de datos

Un equipo que ejecuta MySQL y almacena la base de datos se llama servidor MySQL. Para conectarse a ese servidor se debe verificar si el servidor admite conexiones ya sea remotas o locales y si disponen de las credenciales necesarias para la conexión.

5.4 Clientes y servidores

El servidor MySQL es el servicio *mysqld*, el cual administra los accesos a los datos de MySQL que contienen las bases de datos y tablas.

Para conectarse al servidor MySQL se puede hacer desde la línea de comandos.

\$ mysql

- **-h** nombre del servidor
- **-u** nombre de usuario
- **-p** para preguntar la contraseña de conexión al usuario.

```
$ mysql -h nombre_servidor -u nombre_usuario -p
```

MySQL tiene un esquema de usuarios y privilegios, los usuarios son creados por el administrador con sus respectivos permisos. Tras establecer la conexión, aparecerá el **mysql>**. Para desconectarse solo se debe escribir QUIT o EXIT.

```
$ mysql> QUIT
```

5.5 Tipos de datos

Existen varios tipos de datos para definir a las tablas.

Se pueden clasificar en tres grupos (Casillas et al, 2009:23):

- Numéricos
- Cadena de caracteres
- Fechas y horas

Numéricos

Se utilizan para almacenar todo tipo de datos numéricos, separados por dos categorías, enteros y números con punto flotante.

Todos los tipos numéricos permiten dos opciones:

- **UNSIGNED**. No se pueden almacenar valores negativos.
- **ZEROFILL**. Rellena con ceros a la izquierda los espacios vacíos de la columna. Una columna con el tipo **ZEROFILL** es al mismo tiempo **UNSIGNED** de manera predeterminada y aunque no se especifique.

UNSIGNED y **ZEROFILL** se deben escribir antes que cualquier otro atributo de columna.

Tabla 5.1

Tipos enteros

Tipo	Espacio de almacenamiento	Descripción
tinyint[(M)]	1 byte	Entero muy pequeño.
smallint[(M)]	2 bytes	Entero pequeño.
mediumint[(M)]	3 bytes	Entero mediano.
int[(M)]	4 bytes	Entero.
Bigint[(M)]	8 bytes	Entero grande.

Fuente: *MySQL Documentation: MySQL Reference Manuals*. (1 de Octubre de 2013). Obtenido de <http://dev.mysql.com/doc/>

Tabla 5.2

Numero de punto flotante

Tipo	Espacio de almacenamiento	Descripción
Float	4 bytes	Numero decimal pequeño de simple precisión.
Double	8 bytes	Numero de coma flotante de doble precisión
Decimal	M + 2 bytes	Cadena de caracteres representando un numero flotante

Fuente: *MySQL Documentation: MySQL Reference Manuals*. (1 de Octubre de 2013). Obtenido de <http://dev.mysql.com/doc/>

Cadena de caracteres

Se utilizan para almacenar todo tipo de datos compuesto por caracteres.

Tabla 5.3

Cadena de caracteres

Tipo	Longitud	Descripción
char[(M)]	De 0 a 255 caracteres	Cadena con longitud fija, con relleno de espacios a la derecha.
varchar[(M)]	De 0 a 255 caracteres	Cadena de longitud variable, los espacios en blanco se eliminan al almacenar el valor.
tinytext	Caracteres máximos 255	
Text	Máximo de 65.523 caracteres.	Text y blob son equivalentes pero text respeta las mayúsculas y caracteres acentuados.
Mediumtext	16.777.215 caracteres	
Longtext	4.294.967.295	
enum		Enumeración. Sólo puede tener unos caracteres especificados.
Set		Un conjunto. Puede contener de cero a 64 valores.

Fuente: *MySQL Documentation: MySQL Reference Manuals*. (1 de Octubre de 2013). Obtenido de <http://dev.mysql.com/doc/>

Fechas y horas

Los tipos de datos de fecha y hora son diseñados para trabajar con los datos de tipo temporal y se pueden utilizar para almacenar datos como la hora, día o fechas.

Tabla 5.4

Tipos de datos de fecha y hora

Tipo	Descripción
Date	De 1000-01-01 al 9999-12-31 (AAAA-MM-DD)
Time	00:00:00 a 23:59:59 (HH:MM:ss)
Datetime	De 1000-01-01 al 9999-12-31 (AAAA-MM-DD) 00:00:00 a 23:59:59 (HH:MM:ss)
timestamp[(M)]	Utilizada para valores que contienen partes de fecha y hora tienen un rango de ' 1970-01-01 00:00:01 ' GMT a ' 2038-01-19 03:14:07 ' GMT
year[(M)]	AAAA

Fuente: *MySQL Documentation: MySQL Reference Manuals*. (1 de Octubre de 2013). Obtenido de <http://dev.mysql.com/doc/>

Capítulo 6. Interacciones medicamentosas

6.1 Introducción

La interacción medicamentosa es considerada cuando los efectos de un fármaco son modificados debido a la administración de otro fármaco o alimento.

Las consecuencias se pueden dividir en interacción beneficiosa o adversa, puede resultar en disminución, anulación o aumento del efecto de uno o más fármacos. (Hartshorn, 2006: 112-113).

Se pueden clasificar como físico-químicas, farmacocinéticas e interacción farmacodinamia.

- Las interacciones físico-químicas se consideran cuando dos o más medicamentos interaccionan entre sí, por mecanismo físico químicos.
- La interacción farmacodinamia ocurre cuando existe adición del efecto o antagonismo de los fármacos.
- La interacción farmacocinética ocurre cuando uno de los fármacos es capaz de modificar la absorción, distribución, biotransformación y eliminación del otro fármaco.

6.2 Importancia

El error médico se considera factor de riesgo en relación a la seguridad del paciente. El error médico se define como el fracaso de aplicar completamente un plan de acción como fue propuesto o también del uso de un plan equivocado para alcanzar un objetivo.

El error médico resulta de una equivocación en que no existe mala fe, ni se pone de manifiesto una infracción o imprudencia, como la negligencia, abandono, indolencia, desprecio, incapacidad, impericia e ignorancia profesional.

La medicación es la primera causa sobre incidentes en la seguridad del paciente, estudios realizados en EEUU atribuye a los errores de medicación 7000 fallecimientos anuales. (Kohn et al 1999:18).

6.3 Tipos de interacciones

Cuando se generan las interacciones en una persona, las reacciones pueden ser independientes una de otras; no obstante, el efecto puede ser el aumento de la acción farmacológica o la disminución de ésta. Este tipo de efectos se les conoce como sinergismo y antagonismo.

- Sinergismo

Se llama sinergismo cuando el efecto de una interacción medicamentosa aumenta el efecto de uno o varios de los fármacos que no serían posibles si se administraran solos.

- Antagonismo

Se llama antagonismo cuando el efecto de una interacción medicamentosa disminuye o anula el efecto de uno o varios de los fármacos que se administraron.

6.4 Interacciones con alimentos

Los alimentos tienen la capacidad de inhibir e inducir los efectos de los medicamentos. La interacciones medicamentosas con alimentos y nutrientes no sólo afectan a los fármacos sino que

también a carencias de vitaminas y minerales inducidas por el tratamiento con determinados medicamentos.

Se pueden clasificar de dos formas:

- *Interacciones de los alimentos sobre los medicamentos.*

Este tipo de interacciones es de gran importancia, puesto que actúan sobre la respuesta farmacológica como medio para restaurar la normalidad en el organismo ante una situación patológica y, por tanto, el medicamento no cumpliría con la misión para la que fue prescrito.

- *Interacciones de los medicamentos sobre los alimentos.*

Este tipo de interacciones es de gran importancia, puesto que actúan sobre la respuesta farmacológica como medio para restaurar la normalidad en el organismo ante una situación patológica y, por tanto, el medicamento no cumpliría con la misión para la que fue prescrito. (Samano y Sánchez, 2014).

Capítulo 7. Caso Práctico

7.1 Creación de la Base de Datos

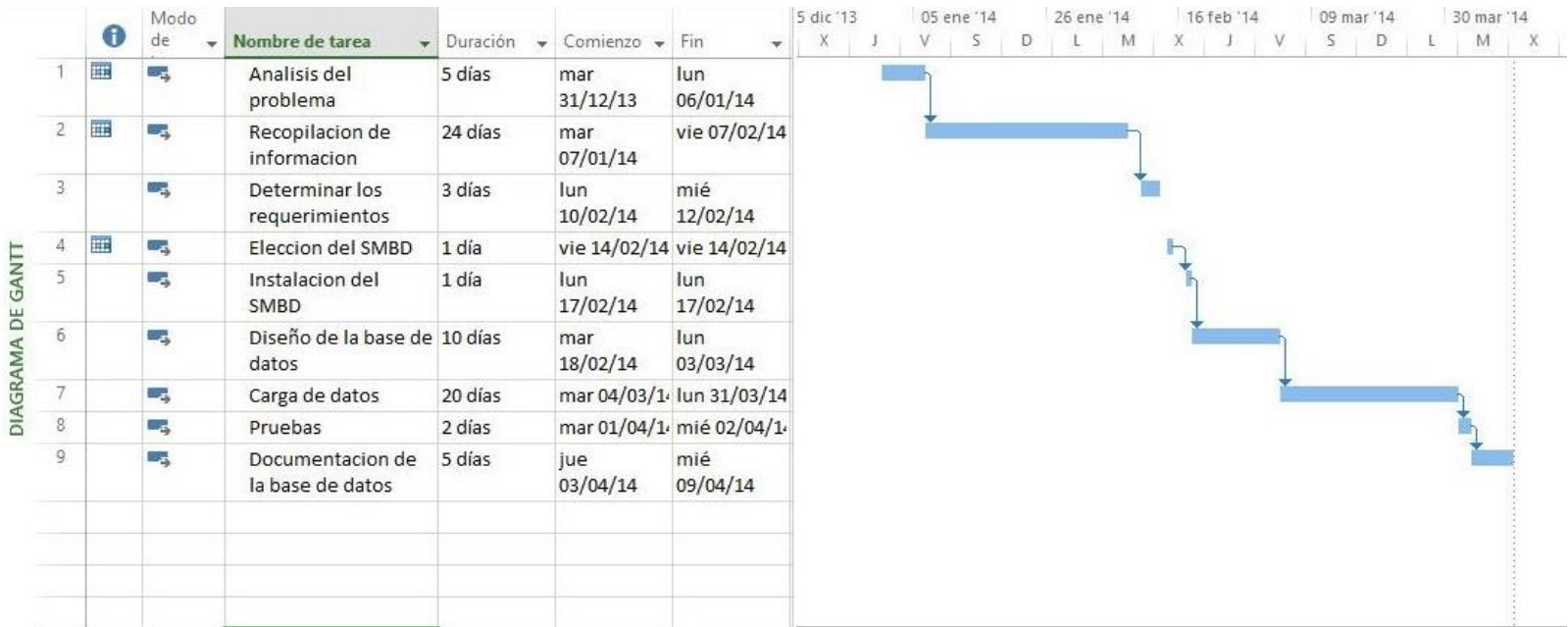
Planificación del proyecto

Para realizar el proyecto se deben planear las actividades para estimar tiempos de las tareas a realizar. La planificación de proyecto se realizó en Project 2013. La calendarización se realizó mediante un diagrama de Gantt (véase figura 7.1).

Figura 7.1

Diagrama de Gantt

Manejador de base de datos



El manejador de base de datos que se utilizó fue MySQL, tomando en cuenta sus características y sus requerimientos para su instalación.

El código generado en MySQL está comentado para futuras modificaciones y actualizaciones.

El equipo donde se creó la base de datos cuenta con las siguientes características:

- Sistema operativo Windows 8.1
- Computadora con procesador Intel i7, memoria RAM de 8 GB, disco duro 1 TB.
- MySQL 5.6

Documentación

En la documentación e investigación del proyecto se utilizó tanto software libre y propietario como Microsoft Office 2013, Notepad++.

La fuente para extraer los datos requeridos fue proporcionada por el Dr. Jesús Ignacio Simón Domínguez, consultor médico de Tecnología Informática Moderna S.A. (TIMSA) los datos se encontraban en forma descriptiva en texto libre.

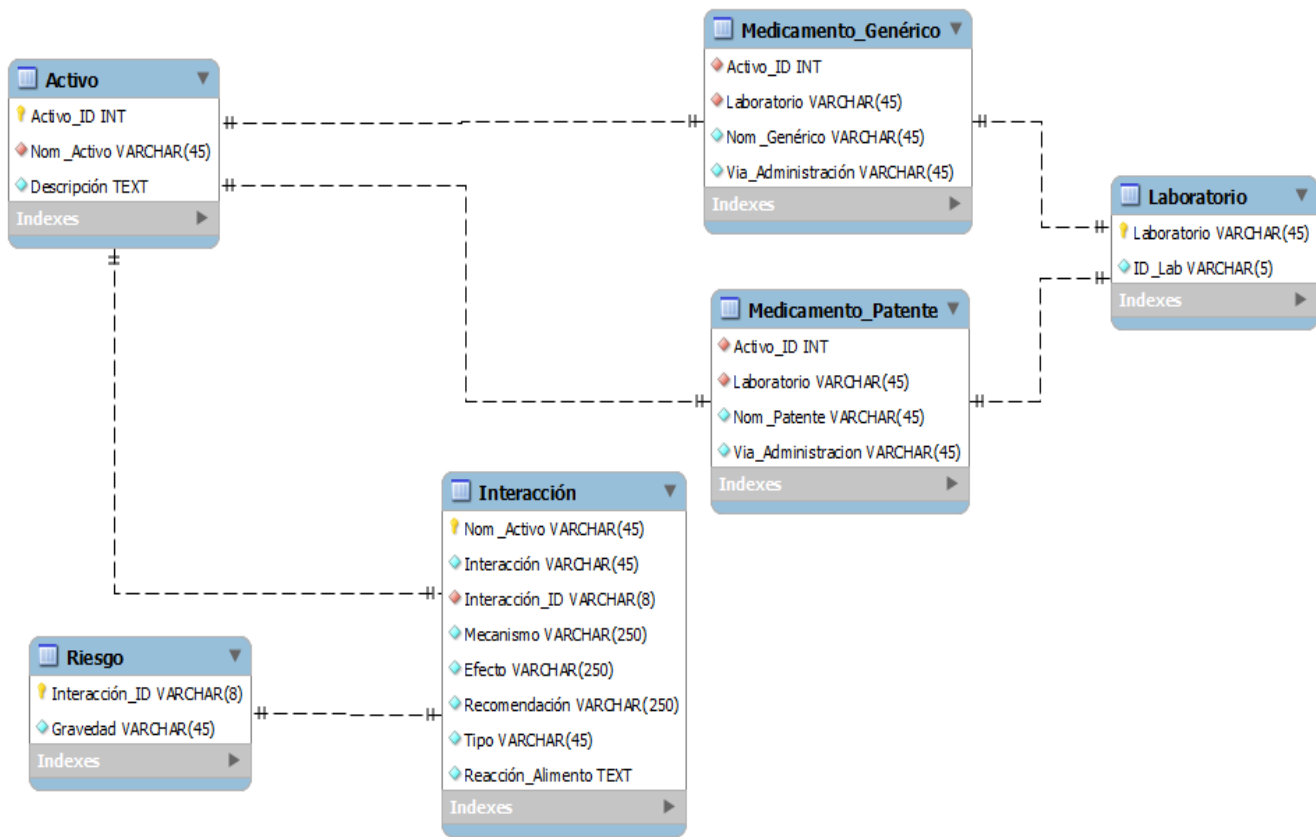
Diseño de la base datos

A continuación se presenta el modelo entidad relación (ver figura 7.2).

Modelo entidad relación.

Figura 7.2

Modelo E/R



Fuente: elaboración propia.

A continuación se presentan las tablas para la conformación de la base de datos que se elaboraron para este proyecto, así como el diccionario de datos, el cual se centra en los datos y la forma que están estructurados.

Tabla 7.1


Descripción de las tablas elaboradas para el diseño de la base de datos

Nombre	Descripción
Activo	Tabla de los principios activos.
Interaccion	Tabla donde se muestran las interacciones entre activos y sus efectos.
Laboratorio	Tabla de los laboratorios.
medicamento_generico	Tabla de los medicamentos genéricos por su principio activo.
medicamento_patente	Tabla de los medicamentos patentes por su principio activo.
Riesgos	Tabla de riesgos entre las interacciones.

Fuente: elaboración propia.

Tabla 7.2
Descripción de tabla principios activos

Columnas

Nombre	Tipo de dato	Descripción
 Activo_ID	INT	Identificador numérico de los activos.
Nom_Activo	VARCHAR(45)	Muestra el nombre del activo.
Descripcion	TEXT	Describe la función del activo.

Fuente: elaboración propia.

Tabla 7.3
Descripción de tabla principios activos

Relaciones


Tabla Primaria	Llave Primaria	Tabla Foránea	Llave Foránea
Activo	Activo_ID	medicamento_generico	Activo_ID
Activo	Activo_ID	medicamento_patente	Activo_ID
Activo	Activo_ID	interaccion	Activo_ID

Fuente: elaboración propia.

Tabla 7.4

Descripción de la tabla interaccion

Descripción: Tabla donde se muestran las interacciones entre activos y sus efectos.

Nombre	Tipo de dato	Descripción
 Interaccion_ID	VARCHAR(8)	Identificador alfa-numérico de las interacciones.
Interaccion	VARCHAR(45)	Nombre del activo que interactúa con el activo principal
Activo_ID	INT	Identificador numérico de los activos.
Mecanismo	VARCHAR(250)	Acción a la interacción.
Efecto	VARCHAR(250)	Efecto de la interacción.
Recomendación	VARCHAR(250)	Recomendación a la interacción.
Tipo	VARCHAR(45)	Describe el tipo de interacción.
Reaccion_Alimento	TEXT	Muestra cuál es la interacción del activo con los alimentos.

Fuente: elaboración propia.

Tabla 7.5

Relaciones de la tabla interaccion


Tabla Primaria	Llave Primaria	Tabla Foránea	Llave Foránea
Interaccion	Interaccion_ID	riesgos	Interaccion_ID
Activo	Activo_ID	interaccion	Activo_ID

Fuente: elaboración propia.

Tabla 7.6

Descripción de la tabla laboratorio

Descripción: Tabla de los laboratorios.

Nombre	Tipo de dato	Descripción
 Laboratorio	VARCHAR(45)	Identificador Nombre de los laboratorios.
Laboratorio_ID	VARCHAR(5)	Id de los laboratorios.

Fuente: elaboración propia.

Tabla 7.7

Relaciones de la tabla laboratorio

Tabla Primaria	Llave Primaria	Tabla Foránea	Llave Foránea
laboratorio	Laboratorio	medicamento_patente	Laboratorio
laboratorio	Laboratorio	medicamento_generico	Laboratorio

Fuente: elaboración propia.

Tabla 7.8

Descripción de la tabla medicamento_generico

Descripción: Tabla de los medicamentos genéricos por su principio activo.

Nombre	Tipo de dato	Descripción
Activo_ID	INT	Identificador numérico de activos.
Laboratorio	VARCHAR(45)	Nombre de los laboratorios.
Nom_Generico	VARCHAR(45)	Nombre del medicamento genérico.
Via_Administracion	VARCHAR(45)	Vía de administración del medicamento.

Fuente: elaboración propia.

Tabla 7.9

Relaciones de la tabla medicamento_generico

Tabla Primaria	Llave Primaria	Tabla Foránea	Llave Foránea
activo	Activo_ID	medicamento_generico	Activo_ID
laboratorio	Laboratorio	medicamento_generico	Laboratorio

Fuente: elaboración propia.

Tabla 7.10

Descripción de la tabla medicamentos_patentes

Descripción: Tabla de los medicamentos patentes por su principio activo.

Nombre	Tipo de dato	Descripción
Activo_ID	INT	Identificador numérico de los activos.
Laboratorio	VARCHAR(45)	Nombre de los laboratorios.
Nom_Patente	VARCHAR(45)	Nombre del medicamento.
Via_Administracion	VARCHAR(45)	Vía de administración del medicamento.

Fuente: elaboración propia.

Tabla 7.11

Relaciones de la tabla medicamentos_patentes

Tabla Primaria	Llave Primaria	Tabla Foránea	Llave Foránea
laboratorio	Laboratorio	medicamento_patente	Laboratorio
activo	Activo_ID	medicamento_patente	Activo_ID

Fuente: elaboración propia.

Tabla 7.12
Descripción de la tabla riesgos

Descripción: Tabla de riesgos entre las interacciones.

Nombre	Tipo de dato	Descripción
Interaccion_ID	VARCHAR(8)	Identificador alfa-numérico de las interacciones.
Gravedad	VARCHAR(45)	Gravedad de las interacciones.

Fuente: elaboración propia.

Tabla 7.13
Relaciones de la tabla riesgos

Tabla Primaria	Llave Primaria	Tabla Foránea	Llave Foránea
interaccion	Interaccion_ID	riesgos	Interaccion_ID

Fuente: elaboración propia.

Base de datos de Interacciones de Medicamentos

El modelo E/R se trasladó a lenguaje SQL que forma parte de MySQL, comenzando con la creación de la base de datos tesis_interacciones y la creación de las tablas.

El motor de almacenamiento con el que se crearon las tablas es Engine InnoDB.

La mayoría de los valores en las tablas son NOT NULL, significando que pueden no quedar vacíos estos campos. Los tipos de datos y su longitud se determinaron por la información que contienen.

A continuación se presentan las capturas de pantalla donde se muestra el código en SQL para la creación y validación de la base de datos elaborada en este proyecto.

Código para la creación de la base de datos y de las tablas Activo, Laboratorio y
Medicamento_Generico

```
mysql> Create database tesis_interacciones;
Query OK, 1 row affected (0.00 sec)

mysql> Use tesis_interacciones;
Database changed
mysql> CREATE TABLE IF NOT EXISTS `TESIS_INTERACCIONES`.`Activo` (
  -> `Activo_ID` INT NOT NULL,
  -> `Nom_Activo` VARCHAR(45) NOT NULL,
  -> `Descripcion` TEXT NOT NULL,
  -> PRIMARY KEY (`Activo_ID`))
  -> ENGINE = InnoDB;
Query OK, 0 rows affected (0.35 sec)

mysql> CREATE TABLE IF NOT EXISTS `TESIS_INTERACCIONES`.`Laboratorio` (
  -> `Laboratorio` VARCHAR(45) NOT NULL,
  -> `Laboratorio_ID` VARCHAR(5) NOT NULL,
  -> PRIMARY KEY (`Laboratorio`))
  -> ENGINE = InnoDB;
Query OK, 0 rows affected (0.46 sec)

mysql> CREATE TABLE IF NOT EXISTS `TESIS_INTERACCIONES`.`Medicamento_Generico` (
  -> `Activo_ID` INT NOT NULL,
  -> `Laboratorio` VARCHAR(45) NOT NULL,
  -> `Nom_Generico` VARCHAR(45) NOT NULL,
  -> `Via_Administracion` VARCHAR(45) NOT NULL,
  -> INDEX `Lab_Generico_idx` (`Laboratorio` ASC),
  -> INDEX `Act_Generico_idx` (`Activo_ID` ASC),
  -> CONSTRAINT `Lab_Generico`
  -> FOREIGN KEY (`Laboratorio`)
  -> REFERENCES `TESIS_INTERACCIONES`.`Laboratorio` (`Laboratorio`)
  -> ON DELETE NO ACTION
  -> ON UPDATE NO ACTION,
  -> CONSTRAINT `Act_Generico`
  -> FOREIGN KEY (`Activo_ID`)
  -> REFERENCES `TESIS_INTERACCIONES`.`Activo` (`Activo_ID`)
  -> ON DELETE NO ACTION
  -> ON UPDATE NO ACTION)
  -> ENGINE = InnoDB;
Query OK, 0 rows affected (0.37 sec)

mysql>
```

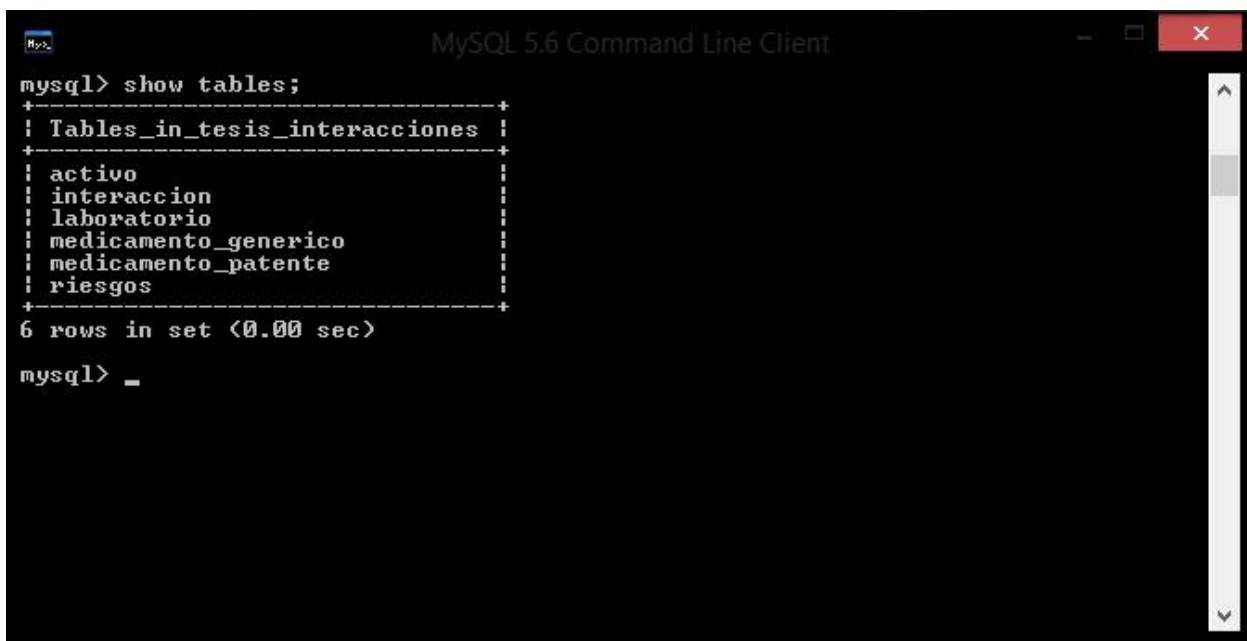
Código para la creación de las tablas Medicamento_Patente e Interaccion

```
mysql> CREATE TABLE IF NOT EXISTS `TESIS_INTERACCIONES`.`Medicamento_Patente` (  
-> `Activo_ID` INT NOT NULL,  
-> `Laboratorio` VARCHAR(45) NOT NULL,  
-> `Nom_Patente` VARCHAR(45) NOT NULL,  
-> `Via_Administracion` VARCHAR(45) NOT NULL,  
-> INDEX `Lab_Patente_idx` (`Laboratorio` ASC),  
-> INDEX `Act_Patente_idx` (`Activo_ID` ASC),  
-> CONSTRAINT `Lab_Patente`  
-> FOREIGN KEY (`Laboratorio`)  
-> REFERENCES `TESIS_INTERACCIONES`.`Laboratorio` (`Laboratorio`)  
-> ON DELETE NO ACTION  
-> ON UPDATE NO ACTION,  
-> CONSTRAINT `Act_Patente`  
-> FOREIGN KEY (`Activo_ID`)  
-> REFERENCES `TESIS_INTERACCIONES`.`Activo` (`Activo_ID`)  
-> ON DELETE NO ACTION  
-> ON UPDATE NO ACTION)  
-> ENGINE = InnoDB;  
Query OK, 0 rows affected (0.45 sec)  
  
mysql>  
mysql> CREATE TABLE IF NOT EXISTS `TESIS_INTERACCIONES`.`Interaccion` (  
-> `Interaccion_ID` VARCHAR(8) NOT NULL,  
-> `Interaccion` VARCHAR(45) NOT NULL,  
-> `Activo_ID` INT NOT NULL,  
-> `Mecanismo` VARCHAR(250) NOT NULL,  
-> `Efecto` VARCHAR(250) NOT NULL,  
-> `Recomendacion` VARCHAR(250) NOT NULL,  
-> `Tipo` VARCHAR(45) NOT NULL,  
-> `Reaccion_Alimento` TEXT NOT NULL,  
-> PRIMARY KEY (`Interaccion_ID`),  
-> INDEX `Activo_Inter_idx` (`Activo_ID` ASC),  
-> CONSTRAINT `Activo_Inter`  
-> FOREIGN KEY (`Activo_ID`)  
-> REFERENCES `TESIS_INTERACCIONES`.`Activo` (`Activo_ID`)  
-> ON DELETE NO ACTION  
-> ON UPDATE NO ACTION)  
-> ENGINE = InnoDB;  
Query OK, 0 rows affected (0.49 sec)  
  
mysql>
```

Código para la creación de la tabla Riesgos

```
mysql> CREATE TABLE IF NOT EXISTS `TESIS_INTERACCIONES`.`Riesgos` (  
-> `Interaccion_ID` VARCHAR(8) NOT NULL,  
-> `Gravedad` VARCHAR(45) NOT NULL,  
-> INDEX `Interaccion_riesgo_idx` (`Interaccion_ID` ASC),  
-> CONSTRAINT `Interaccion_riesgo`  
-> FOREIGN KEY (`Interaccion_ID`)  
-> REFERENCES `TESIS_INTERACCIONES`.`Interaccion` (`Interaccion_ID`)  
-> ON DELETE NO ACTION  
-> ON UPDATE NO ACTION)  
-> ENGINE = InnoDB;  
Query OK, 0 rows affected (0.44 sec)
```

Tablas existentes en la base de datos tesis_interaccion



```
MySQL 5.6 Command Line Client  
mysql> show tables;  
+-----+  
| Tables_in_tesis_interacciones |  
+-----+  
| activo  
| interaccion  
| laboratorio  
| medicamento_generico  
| medicamento_patente  
| riesgos  
+-----+  
6 rows in set (0.00 sec)  
mysql> _
```

Carga de datos

Para la carga de datos, la sentencia utilizada fue la siguiente:

```
INSERT INTO nom_tabla VALUES (valor_columna1, valor_columna2, valor columna3);
```

Los datos insertados fueron previamente definidos y validados por el Dr. Jesús Simón Domínguez.

7.2 Pruebas y validación

Para la realización de pruebas del sistema se crearon consultas para detectar fallas y hacer las modificaciones necesarias.

Las consultas se realizaron con la herramienta grafica de MySQL Workbench para mejor visualización. Se utilizó un 20% de los datos para estas pruebas.

Como un ejemplo de las pruebas realizadas del funcionamiento de la base de datos, en la siguiente figura se presenta la consulta a la base de datos que devuelve la interacción de medicamentos que contengan alcohol.

La validación de la base de datos fue realizada por el Dr. Jesús Ignacio Simón Domínguez, asesor de la empresa TIMSA, la cual proporcionó la información sobre los medicamentos y sus interacciones, que se utilizó como fuente para el análisis y diseño de una base de datos para la mitigación de riesgos derivados por las interacciones medicamentosas en hospitales.

Figura 7.3

Pantalla de consulta de medicamentos que contienen alcohol

The screenshot shows a SQL query editor window with the following SQL code:

```
1 SELECT activo.Activo_ID, activo.Nom_Activo, interaccion.Interaccion, interaccion.efecto, interaccion.mecanismo, interaccion.Reaccion_Alimento
2 FROM activo
3 INNER JOIN interaccion
4 ON activo.Activo_ID=interaccion.Activo_ID
5 where interaccion.Interaccion='alcohol';
6
```

Below the query editor is a 'Result Grid' with the following data:

Activo_ID	Nom_Activo	Interaccion	efecto	mecanismo	Reaccion_Alimento
1	Abacavir	Alcohol	La admin...	No descrito	Los alimentos no modifican la absorción de abacavir por lo que puede administrarse con ellos.

Conclusiones

En relación al objetivo general del proyecto, éste se cumplió dado que se analizó y diseñó una base de datos que ayude en la correcta toma de decisiones de médicos y enfermeras. Dicha base de datos facilita que se conozcan los efectos secundarios que pudieran surgir del uso de medicamentos y sus interacciones y además alerta sobre combinaciones que deberán ser descartadas para evitar serias consecuencias. Asimismo, facilita la prescripción de medicamentos, describiendo los tipos de interacciones entre medicamentos y alimentos.

Durante la realización del proyecto, se comprobó la funcionalidad del manejador de base de datos MySQL por sus características de facilidad de uso, su flexibilidad en diferentes sistemas operativos, sencillo de descargar, instalar y configurar, su libre distribución bajo los términos de la licencia GPL, su grado de estabilidad y rápido desarrollo, lo cual quedo comprobado al realizar este proyecto.

Utilizando MySQL, se ha diseñado y elaborado una base de datos con algunos de los activos que encontramos en los medicamentos, así como las interacciones entre ellos y sus principales riesgos que resultan de las mismas.

El proyecto contribuye a tratar de mitigar un riesgo siempre existente cuando se prescriben medicamentos. La recopilación de todos los datos y su disponibilidad en un manejador de base de datos para el personal médico, resulta útil para reconocer una posible reacción o una detección de una interacción medicamentosa, permitiendo analizar, controlar y actualizar la información de forma periódica, mostrando alertas, gravedad de la interacción, descripción del mecanismo y recomendación para realizar la prescripción de los medicamentos por el médico.

Lo anterior contribuye a preservar la salud de los seres humanos, lo cual es fundamental para el desarrollo sustentable de nuestro país.

Con este proyecto he tenido la oportunidad de seguir aprendiendo los conocimientos que adquirí durante mi estancia como alumno en la Facultad, dejándome una rica experiencia para mi desarrollo profesional.

Bibliografía

- APRENDE SQL. (Consultado el 8 de Septiembre de 2013). Obtenido de http://books.google.com.mx/books?id=FR3ZfyrY8_gC&printsec=frontcover&dq=SQL&hl=es419&sa=X&ei=JsESUr7hGoaV2QWw4IHgAQ&ved=0CDAQ6AEwAA#v=onepage&q=SQL&f=false
- Camps Paré, Rafael. (2009). Introducción a las bases de datos. España: OUC.
- Casillas Santillán, L. A., Gibert Ginestá, M., & Pérez Mora, O. (2009). Bases de Datos en MySQL 1. España: OUC.
- Codd, E.F. (1985a). Is Your DBMS Really Relational? Computerworld. Octubre 14.
- Codd, E.F. (1985b). Does Your DBMS Run by the Rules? Computerworld. Octubre 21.
- Conceptos básicos del diseño de una base de datos. (Consultado el 28 de Agosto de 2013). Obtenido de <http://office.microsoft.com/es-mx/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx>
- Connolly, T., & Begg, C. (2005). Sistemas de bases de datos: Un enfoque práctico para diseño, implementación y gestión. España: Pearson.
- Coronel, C. y Rob P., (2003). Sistemas de bases de datos: diseño, implementación y administración. México: Thomson.
- Costal Costa, Dolors. (2009). El modelo relacional y el álgebra relacional. España: OUC.
- Costal Costa, Dolors. (2009). Introducción al diseño de bases de datos. España: OUC.
- Date, C. J. . (2001). Introducción a los Sistemas de Bases de Datos. México: Pearson Educación.
- El modelo de datos entidad-relación. (Consultado el 3 de Marzo de 2013). Obtenido de <http://www.cs.us.es/cursos/bd-2005/HTML/modeloER.htm>
- Etapas del diseño de bases de datos. (Consultado el 11 de Febrero de 2014). Obtenido de <http://www.dataprix.com/11-etapas-diseno-bases-datos>
- García Arnao, Odalys; Alfonso Orta, Ismary; García Orihuela, Marlene; González Valcárcel, Lourdes. (Consultado el 11 de Mayo de 2014). "Identificación de Problemas Relacionados con Medicamentos (PRM) en. GERONINFO. PUBLICACIÓN DE GERONTOLOGÍA Y GERIATRÍA, 2.

Obtenido de Identificación de Problemas Relacionados con Medicamentos (PRM) en:
http://www.sld.cu/galerias/pdf/sitios/gericuba/identificacion_de__pmr.pdf

- Gilfillan, Ian. (2003). La Biblia de MySQL . España: Anaya Multimedia.
- Hartshorn E.A. (2006). Evolution of Drug–Drug Interactions: A Personal Viewpoint. The Annals of Pharmacotherapy.
- History of SQL. (Consultado el 16 de Enero de 2014). Obtenido de <http://ccollins.wordpress.com/2007/05/20/history-of-sql/>
- Informática Cliente-Servidor. (Consultado el 23 de Septiembre de 2013). Obtenido de http://www.angelfire.com/my/jimena/so2/com_guia2.htm
- Interacciones medicamentosas potenciales en pacientes de una unidad de terapia intensiva de un hospital universitario. (Consultado el 23 de Septiembre de 2013). Obtenido de http://www.scielo.br/pdf/rlae/v17n2/es_13.pdf
- Kohn, T., L., Corrigan , J., & Donaldson, M. (1999). To Err is Human: Building a Safer Health System. Washington, D.C.: The National Academies Press.
- Leape L., Brennan A., T., Laird, N., Lawthers, A., Localio, R., Barnes, B., . . . Hiatt, H. (1991). The Nature of Adverse Events in Hospitalized Patients Results of the Harvard Medical Practice Study II. The NEW ENGLAND JOURNAL of MEDICINE, 324-384.
- Limits of Table Column Count and Row Size. (Consultado el 25 de Enero de 2014). Obtenido de <https://dev.mysql.com/doc/refman/5.0/en/column-count-limit.html>
- Martín Escofet, Carmem. (2009). El Lenguaje SQL. España: OUC.
- Mitchel Pierre, Aristil Chéry. (2010). Manual de Farmacología Básica y Clínica 5 Ed. México: Mc Graw Hill.
- Modelo Entidad- Relación. (Consultado el 4 de Octubre de 2013). Obtenido de <http://cursos.aiu.edu/Base%20de%20Datos/pdf/Tema%203.pdf>
- MySQL Documentation: MySQL Reference Manuals. (Consultado el 1 de Octubre de 2013). Obtenido de <http://dev.mysql.com/doc/>
- Nevado Cabello, Ma. Victoria. (2010). Introducción a las Bases de Datos Relacionales. España: Vision Libros.
- Piattini, M., & Castaño, A. (1999). Fundamentos y modelos de Bases de Datos 2. México: Alfaomega.

- Raffa, B., R., M. Rawls, S., & Portyansky Beyzarov, E. (2005). Netter Farmacologia Ilustrada. España: Elsevier Masson.
- Ramez Elmasri y Shamkant B. Navathe. (2007). Fundamentos de Sistemas de Bases de Datos. España: Pearson.
- Sámano San Miguel, MT, Sánchez Méndez, JL. (Consultado el 04 de Junio de 2014). Información Terapeutica del Sistema Nacional de Salud. Obtenido de Interacciones alimento/medicamento. Obtenido de https://www.msssi.gob.es/biblioPublic/publicaciones/recursos_propios/infMedic/docs/vol35_1_Interacciones.pdf
- Sentencias de manipulación de datos. (Consultado el 1 de Octubre de 2013). Obtenido de <https://dev.mysql.com/doc/refman/5.0/es/data-manipulation.html>
- SQL Tutorial. (Consultado el 11 de Abril de 2014). Obtenido de <http://www.w3schools.com/sql/default.asp>
- United Nations [UN], 1987. Report of the World Commissions on Environment and Development: Our Common Future, Oslo, disponible en <http://www.un-documents.net/our-common-future.pdf>, (01 Diciembre 2013).