UNIVERSIDAD NACIONAL AUTONOMA
DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

CAMPUS ARAGÓN

“INFORME DEL DIPLOMADO EN ADMINISTRACIÓN DE BASE DE DATOS Y DEL PROYECTO DEL SISTEMA DE ADMINISTRACIÓN VÍA WEB”

SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÒN PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

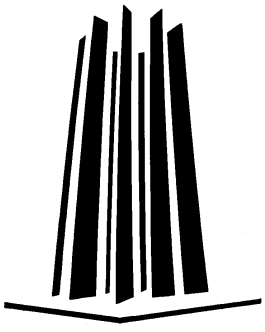
INGENIERO EN COMPUTACIÓN

PRESENTA:

CAMPUZANO HERNÁNDEZ OSCAR

ASESOR:

ING. SILVIA VEGA MUYTOY



MEXICO 2008

D e d i c a t o r i a

Dedico este trabajo muy en especial a mis padres, a ellos que siempre me han apoyado en todo momento a lo largo de mi vida, ellos que con su amor, cariño y ejemplo me han hecho crecer como persona.

Sin su apoyo no hubiera podido conseguir lo que hasta ahora he hecho, además me han enseñado a luchar a librar los obstáculos que se van presentando a lo largo de la vida, me han enseñado lo valioso de la vida, de hecho desde que ellos me pusieron el nombre me bendijeron sin ni siquiera ellos saberlo, ya que mi nombre significa “Guerrero que sabe saltar obstáculos”.

En especial quiero agradecer a mi papá ya que el con su ejemplo y sus consejos me han enseñado a ver de una manera muy diferente la vida, siempre he querido ser como él, ya que siempre lo he admirado, el nunca se da por vencido en la vida, para el no hay imposibles, siempre trata de ser mejor persona día a día.

También agradezco a mis hermanos que me han apoyado en todo lo que me propongo y siempre me han alentado para seguir adelante y nunca quedarme estancado, les agradezco por sus consejos y regaños.

Un agradecimiento muy especial a mis primas “Angy, Ely, Peque, y Vicky“ y a toda su familia ya que me dejaron ser y sentirme parte de ellos. Ellas con su cariño, comprensión, confianza y consejos me ayudaron a salir adelante en una etapa de mi vida en la que no la estaba pasando bien y quería abandonar todo y ya no continuar.

Me enseñaron también a ver la vida desde diferentes perspectivas, a gozar cada minuto de ella, de enfrentar los problemas que se van presentando día a día y siempre esforzarme y dar el máximo en todo momento.

Mis amigos también han sido parte fundamental para lograr esta meta que me trace, ellos siempre me alientan a seguir adelante, me apoyan y aconsejan y se que siempre cuento con ellos cuando los necesito.

Una mención especial se merece mi amigo y hermano del alma “Cesar Eduardo Elizondo Herrera” que aunque él no esta más con nosotros también fue y seguirá siendo parte fundamental en mi vida, gracias por toda tu amistad que me diste y estoy tratando de cumplir un sueño que los dos teníamos, de ti aprendí muchísimas cosas, y ahora se que me estas apoyando en este y en cada momento de mi vida desde donde estas ya que para mi te haz vuelto mi ángel de la guarda.

Gracias a todos los que han aportado un granito de arena en mi vida, ya que me ha servido muchísimo para crecer como persona, que Dios los bendiga a cada uno de ustedes que ha dejado una enseñanza en mí.

“El que no vive para servir no sirve para vivir.”

INDICE

INTRODUCCIÓN

La gran diversidad, volumen e importancia tanto económica como estratégica, que tiene la información en cualquier organización hoy en día, ya sea institución pública o privada, implica el uso de sistemas manejadores de bases de datos para garantizar su seguridad, consistencia, integridad, accesibilidad, entre otros factores.

Esto ha creado la necesidad de contar con profesionales de las tecnologías de la información, los cuales tengan los conocimientos y habilidades requeridas para administrar en forma eficaz y eficiente, tan valioso recurso como lo es la información.

El informe tiene como objetivos principales los siguientes puntos:

- Proporcionar los conocimientos teórico – prácticos que permitan administrar bases de datos en forma eficaz y eficiente.
- Identificar y llevar a la práctica las diversas tareas de un administrador, así como los mecanismos de seguridad, control y puesta a punto de los servidores.
- Conocer las principales características de los Sistemas Manejadores de Bases de Datos, tanto comerciales, como libres y utilizar el SQL ANSI 89, 92, y 99 para la definición, control y consulta de datos.
- Creación de un sistema Web para la administración de usuarios, bases de datos y creación de respaldos de una manera más fácil e intuitiva, a través de un sistema Web.

Este informe conlleva a adquirir conocimientos para aprender los principales objetivos que tienen que llevar acabo los administradores de base de datos para realizar su tarea de una forma más eficiente, además de que se adquieren conocimientos extras para complementar el trabajo del administrador de la base de datos, y esto se hace para que se tenga un panorama mayor de las tecnologías con las que tiene que interactuar el manejador de base de datos.

Como ya se menciono la información para una empresa es de vital importancia, ya que ella ayuda a la toma de decisiones, ya sea para implementar una nueva estrategia de venta ó para que basado en ella se pueda invertir en otros puntos que la empresa no haya considerado.

Además de que la implementación de todos los conocimientos adquiridos se ve en el segundo capitulo donde se toman todas las herramientas aprendidas y se desarrolla un caso práctico el cual nos ayuda a la administración de nuestro manejador de base de datos que hemos escogido, y para esto se construye un sistema Web para que de una forma más fácil se pueda administrar nuestro sistema sin tener que aprendernos las sentencias de los comandos para la administración de los usuarios, bases de datos, respaldos, etc.; basta con utilizar la interfaz que es más grafica e intuitiva.

CAPITULO 1.- INFORME DEL DIPLOMADO DE ADMINISTRACIÓN DE BASE DE DATOS.

1.1.-SISTEMAS DE INFORMACIÓN Y EL MODELO DE DATOS RELACIONAL.

1.1.1 ¿Qué es un dato?

Un dato es la unidad mínima de información, aunque también es un hecho aislado sin evaluar o un valor que por si solo no significa nada.

Esto quiere decir que no tiene ningún contexto en el cual se pueda interpretar, un ejemplo podría ser cuando se ve un número, éste a simple vista no dice nada, se necesita conocer a que se refiere ese número, es decir, conocer el contexto para que ese dato se pueda procesar y tenga algún significado para nosotros.

1.1.2 ¿Qué es la información?

La información es un conjunto de datos interrelacionados entre sí, que tienen un significado del cual se puede obtener conocimientos para una futura toma de decisiones. Esto es importante para las personas que poseen la información ya que basados en ella se pueden tomar medidas para su aprovechamiento.

Como se dijo anteriormente un dato es la unidad mínima de información y la información se obtiene asociando los hechos en un contexto determinado, es decir, la adición o el procesamiento de los datos proporcionan el conocimiento o entendimiento de ciertos factores.

Otra forma en la cual se puede definir a la información seria que ésta es un acontecimiento o una serie de acontecimientos, que llevan un mensaje y que al ser percibida por el receptor mediante alguno de sus sentidos, amplía sus conocimientos, en esta relación sólo el destinatario puede evaluar el significado y utilidad de la información recibida.

La información se ha convertido en la parte vital para las empresas hoy en día, ya que ahora el que tiene la información tiene el poder, además de que la información nos sirve para tomar decisiones en base a ella, para así poder aprovechar sus beneficios de una mejor forma.

1.1.3 Análisis de la información.

El análisis de la información es un modelo de datos que consiste en la representación conceptual de la problemática que se desea resolver y cuya característica primordial es la claridad de su contenido. Esta definición se puede interpretar como que se necesita hacer un balance entre el costo de la información y el beneficio que se logra con ésta.

La información es un recurso sumamente valioso para cualquier organización, sin embargo, la obtención de la información formal genera gastos y su valor sólo puede ser comparado con el valor que tendrá para el receptor final.

Algo importante de mencionar es que el costo de producción de la información es tangible y se puede medir gracias a los dispositivos y medios utilizados, pero la información es conceptual por naturaleza y no tiene características tangibles salvo representaciones simbólicas.

Por lo tanto se pueden definir algunas características con los que cuenta la información para que en base a estos se pueda dar una idea del valor de la información.

Características del valor de la información.

- Accesible: Es la facilidad y rapidez con que se obtiene la información resultante.
- Clara: Se refiere a la integridad y entendimiento de la información sin ambigüedades.
- Precisa: Que sea lo más exacta posible.
- Propia: Debe de haber relación entre el resultado y lo solicitado por el usuario.
- Oportuna: Menor duración del ciclo (entrada, procesamiento y entrega al usuario).
- Flexible: Adaptabilidad de la información a la toma de decisiones.
- Verificable: Que se pueda examinar la información.
- Imparcial: No se puede alterar o modificar la información (sólo por el dueño).
- Cuantificable: Todo dato procesado – produce información.

1.1.4 Definición de una base de datos.

Una definición general podría ser que una base de datos es un conjunto de datos almacenados y relacionados entre sí con un objetivo común.

Aunque se podría realizar una definición un poco más especializada en la cual se podría decir que una base de datos es una colección de datos integrados, con redundancia[1] controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real.

Pero una definición más aceptable podría ser que una base de datos puede ser considerada como un conjunto de datos persistentes[2] que pertenecen al mismo contexto, almacenados sistemáticamente bajo un modelo los cuales sirven para su uso posterior en la toma de decisiones.

Tipos de Bases de Datos.

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

- Según la variabilidad de los datos almacenados:
  - Bases de datos estáticas: Son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, para así realizar proyecciones y tomar decisiones.
  - Bases de datos dinámicas: Son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta.
- Según el contenido: Bases de datos bibliográficas, bases de datos numéricas, bancos de imágenes audio, video, multimedia, etc.

1.1.5 Arquitectura para bases de datos.

Hace unos años atrás, las bases de datos eran el resultado de una compleja programación y de complicados mecanismos de almacenamiento. Con la popularización de la informática, la aparición de aplicaciones

[1] La redundancia se refiere a la duplicidad de datos no necesarios en la base de datos.
[2] Un dato persistente es aquel que es recuperado en un futuro independientemente de la aplicación o programa que lo creo.

específicas también trajo con ella la disponibilidad de herramientas de gestión de datos que dieron lugar a los denominados Sistemas de Gestión de Bases de Datos, identificados por sus siglas SGBD por su acrónimo en ingles DBMS (Data Base Management Systems).

De esta manera, la gestión de las bases de datos pudo liberarse de las grandes computadoras centrales para distribuirse según los intereses de los usuarios dotando de autonomía en la gestión de información a muchas entidades.

Para dotar a los usuarios de cierta seguridad en el intercambio de datos entre diferentes sistemas y en el diseño de archivos y bases de datos, fue necesario normalizar los esquemas que guiaban la creación de las bases de datos.

Un sistema de base de datos se encuentra dividido en módulos cada uno de los cuales controla una parte de la responsabilidad total de sistema. En la mayoría de los casos, el sistema operativo proporciona únicamente los servicios más básicos y el sistema de la base de datos debe partir de esa base y controlar además el manejo correcto de los datos.

El diseño de un sistema de base de datos debe incluir la interfaz[3] entre el sistema de base de datos y el sistema operativo. Los componentes funcionales de un sistema de base de datos, son los siguientes:

- Gestor de archivos: Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar la información.
- Manejador de base de datos: Sirve de interfaz entre los datos y los programas de aplicación.
- Procesador de consultas: Traduce las proposiciones en lenguajes de consulta a instrucciones de bajo nivel. Además convierte la solicitud del usuario en una forma más eficiente.
- Compilador de DDL: Convierte las proposiciones DDL en un conjunto de tablas que contienen meta datos, éstas se almacenan en el diccionario de datos.
- Archivo de datos: En él se encuentran almacenados físicamente los datos de una organización.
- Diccionario de datos: Contiene la información referente a la estructura de la base de datos.
- Índices: Permiten un rápido acceso a registros que contienen valores específicos.

Las bases de datos respetan la arquitectura de tres niveles definida, para cualquier tipo de base de datos, por el grupo ANSI/SPARC[4]; el objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física.

El SGBD (DBMS) es el software que maneja todos los accesos a la base de datos, cada solicitud de acceso de un usuario al sistema es interpretada e inspeccionada generando a continuación una respuesta coherente a las necesidades de la pregunta. La interfaz con el usuario es el límite de acceso que tiene un usuario común a la base, todo lo que está bajo este límite es transparente o desconocido para él.

[3] Es la forma en que el usuario se puede comunicar con la computadora.
[4] ANSI/SPARC es un grupo de normalización creado en 1969 para estudiar el impacto de los S.G.B.D. en los sistemas de información y cuyos resultados, publicados en 1975 propusieron el uso de tres niveles de descripción de datos.

Un esquema más genérico que permite ilustrar los niveles existentes dentro de la arquitectura de una base de datos es el de la fig. 1.1.

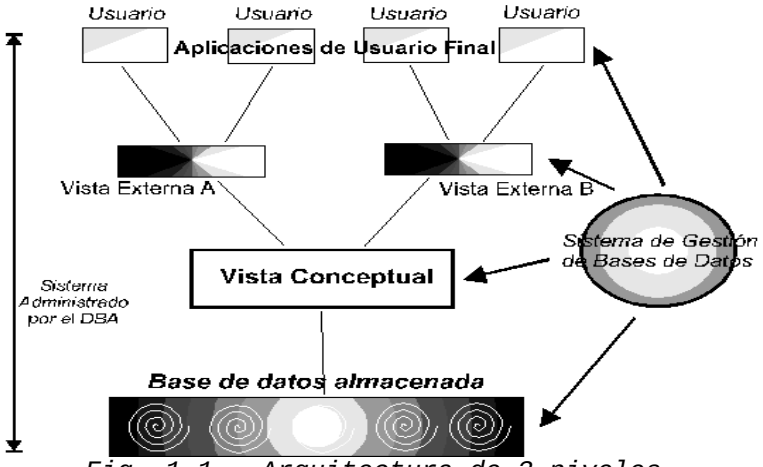


Fig. 1.1.- Arquitectura de 3 niveles.

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que se puede definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se pueden definir dos tipos de independencia de datos:

- *Independencia lógica*: Es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o reducirla. Por ejemplo, si se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberían verse afectados.

- *Independencia física*: Es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos).Por ejemplo, puede ser necesario reorganizar ciertos archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la lógica.

1.1.6 Características de las bases de datos.

Existen ciertas características con las que cuentan las bases de datos, entre ellas se encuentran las siguientes:

Redundancia

La redundancia de datos se refiere, a la existencia de información repetida o duplicada no necesaria en diferentes tablas dentro de una base de datos.

Dentro de una base de datos relacional la redundancia debe ser mínima y controlada ya que si no es así puede conducir a problemas con la integridad y consistencia de los datos. En ocasiones existirán motivos válidos de negocios o técnicos para mantener varias copias de los mismos datos almacenados.

Consistencia

Frecuentemente los problemas de consistencia de datos se deben a la redundancia de éstos.

Es muy probable que surjan incongruencias al almacenar la misma información en más de un lugar; ya que al modificar, eliminar o agregar un dato, en esas condiciones, debe realizarse en cada una de las instancias del mismo con el riesgo de no realizarlo en su totalidad, generando en este caso datos inconsistentes.

Integridad

La integridad de una base de datos se refiere no sólo a que los datos sean consistentes dentro de la base, sino además, que los valores que posean los datos sean válidos de acuerdo a las dependencias funcionales entre tablas y de acuerdo a las políticas de negocio.

Un ejemplo de esto puede ser que por ejemplo al introducir el sexo de un empleado no establezcamos bien las restricciones y podamos introducir de manera indiscriminada otro dato que no cumple con las reglas.

La integridad de la base de datos se puede lograr mediante:

- El mantenimiento de una redundancia mínima y controlada.
- El establecimiento de llaves primarias o índices primarios.
- La creación de reglas de validación durante la inserción y edición de datos.

Seguridad

Hoy en día se considera a la información de una empresa como uno de los activos más valiosos e importantes, por lo que la seguridad de la misma es muy importante.

La seguridad implica asegurar que los usuarios están autorizados para llevar acabo lo que tratan de hacer.

La seguridad de una base de datos se refiere principalmente al control de acceso, modificación y definición, tanto de los datos como de la estructura de la base de datos por parte de los diferentes usuarios a la misma.

Por otro lado, una base de datos debe cumplir con las siguientes condiciones:

- Los datos han de estar almacenados juntos.

- Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento ya que lo importante para estos usuarios es la información contenida.

- Los datos son compartidos por diferentes usuarios y programas de aplicación; existe un mecanismo común para inserción, actualización, borrado y consulta de los datos.

- Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos.

- Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes SGBD.

1.1.7 Modelo de datos.

Modelo: Es una representación de la realidad que contiene las características generales de algo que se va a realizar. En base de datos, esta representación se elabora de forma gráfica.

Modelo de datos: Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

Los modelos de datos se dividen en tres grupos:

- Modelos lógicos basados en objetos.
- Modelos lógicos basados en registros.
- Modelos físicos de datos.

Modelos lógicos basados en objetos

Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo se representan los datos de tal forma como son captados en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente.

El modelo más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación.

Modelo Entidad-Relación

Denominado por sus siglas como E-R; este modelo representa a la realidad mediante *entidades* que son objetos que existen y que se distinguen de otros por sus características, por ejemplo, un alumno se distingue de otro por sus características particulares como lo es el nombre o el número de cuenta asignado al entrar a una institución educativa.

Modelos lógicos basados en registros

Se utilizan para describir datos en los niveles conceptual y físico.

Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los tres modelos de datos más ampliamente aceptados son:

- Modelo jerárquico.
- Modelo de red.
- Modelo relacional.

Modelo jerárquico

Es similar al modelo de red en cuanto a las relaciones y datos, ya que estos se representan por medio de registros y sus ligas. La diferencia radica en que están organizados por conjuntos de árboles en lugar de gráficas arbitrarias.

En este tipo de modelos la organización se establece en forma de árbol, donde la raíz es un nodo ficticio. Así se tiene que, una base de datos jerárquica es una colección de árboles de este tipo mostrada en la fig. 1.2.

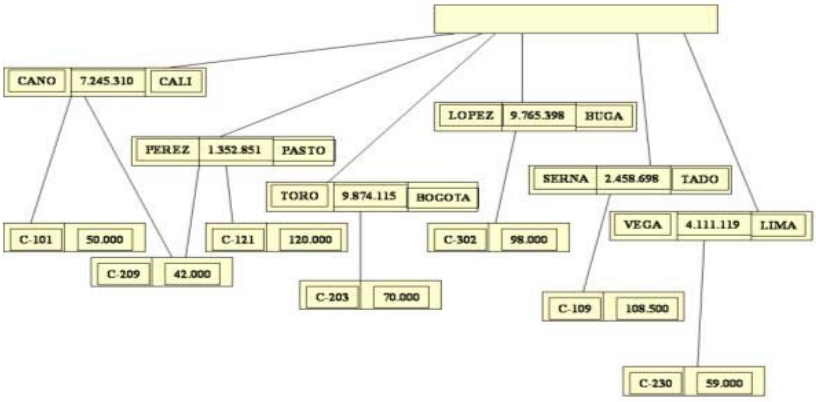


Fig. 1.2.- Modelo Jerárquico.

Modelo de red

Este modelo representa los datos mediante colecciones de registros, sus relaciones se representan por medio de ligas o enlaces, los cuales pueden verse como punteros.

Un *registro* es una colección de campos (atributos), cada uno de los cuales contiene solamente almacenado un valor, el *enlace* es la asociación entre dos registros exclusivamente, así que se puede ver como una relación estrictamente binaria, así como se muestra en la fig. 1.3.

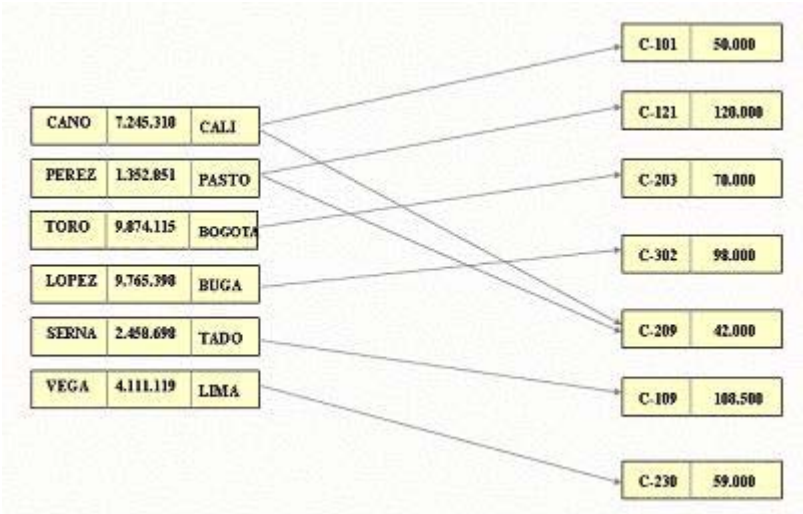


Fig. 1.3.- Modelo de Red.

Modelo relacional

En este modelo se representan los datos y las relaciones entre éstos, a través de una colección de tablas, en las cuales los renglones (tuplas) equivalen a cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la tupla, por ejemplo, para las tablas *cliente* y cuenta, se tendría un ejemplo como se puede apreciar en la fig. 1.4.

CLIENTE

NOMBRE	CEDULA	CUENTA	CIUDAD
CANO	7.245.310	C-101	CALI
PEREZ	1.352.851	C-121	PASTO
TORO	9.874.115	C-203	BOGOTA
LOPEZ	9.765.398	C-302	BUGA
SERNA	2.458.698	C-109	TADO
VEGA	4.111.119	C-230	LIMA
CANO	7.245.310	C-209	CALI
PEREZ	1.352.851	C-209	PASTO

CUENTA

CUENTA	SALDO
C-101	50.000
C-121	120.000
C-203	70.000
C-302	98.000
C-209	42.000
C-109	108.500
C-230	59.000

Fig. 1.4.- Modelo Relacional.

Existen dos formas de representar las relaciones; sin embargo, es necesario definir el concepto de llave primaria, que no es más que un campo que se define como atributo principal y que de una manera u otra es una forma única para identificar a una entidad.[5]

Para ejemplificar esto se diría que la cuenta en la tabla *cuenta* se distingue de otro porque las cuentas no pueden ser iguales, y en la tabla *cliente* el campo cuenta sólo es un atributo más de la tabla y en éste caso se pueden repetir los valores.

Modelos físicos de datos

Se usan para describir a los datos en el nivel más bajo, aunque existen muy pocos modelos de este tipo, básicamente capturan aspectos de la implementación de los sistemas de base de datos. Existen dos clasificaciones de este tipo que son:

- Modelo unificador
- Memoria de elementos.

1.1.8 Arquitectura Cliente / Servidor.

Se puede entender el término cliente-servidor como un sistema en el que una máquina cliente solicita a una segunda máquina llamada servidor que ejecute una tarea específica, el cliente suele ser una PC conectada a una red LAN y el servidor, como un servidor de archivos PC o un servidor de archivos UNIX.

El modelo Cliente / Servidor como se puede apreciar en la fig. 1.5 se define como la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o a cualquier otro recurso dentro del grupo de trabajo y/o a través de la empresa en diferentes plataformas.

Esta arquitectura se conforma de dos tipos componentes que se comunican a través de una red: La red proporciona la conexión entre los clientes y los servidores.

- Back-end: Servidor
- Front-end: Cliente

[5] Una entidad es aquella que representa un objeto del mundo real como por ejemplo la tabla cliente del modelo relacional.

El Servidor procesa las peticiones que hacen los clientes, y cuando es posible regresa el resultado.

El Cliente envía peticiones al servidor y manipula las respuestas. Éste puede: desplegar y manejar el ambiente de trabajo de la aplicación y la interfaz de usuario, llevar a cabo la validación de datos, desplegar reportes y representar datos gráficamente.

La secuencia de eventos cuando un usuario accede al servidor de bases de datos se puede generalizar en los siguientes pasos.

1. El usuario crea su consulta sobre los datos.
2. El cliente formatea la consulta en lenguaje SQL y la envía a través de la red.
3. El servidor de base de datos verifica los permisos sobre los datos a consultar.
4. El servidor de base de datos procesa la consulta y regresa los resultados.
5. El cliente recibe la respuesta y la presenta al usuario.
6. El usuario visualiza y manipula los datos y reinicia el proceso.

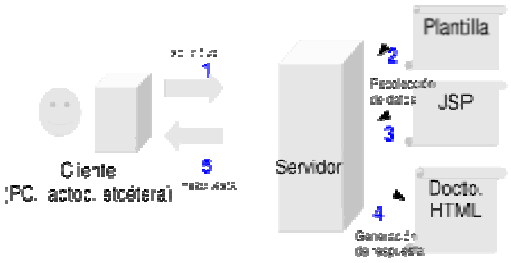


Fig. 1.5.- Modelo Cliente/Servidor.

1.1.9 Modelo Relacional.

La introducción por Codd, muy a finales de los sesenta, de la teoría de las relaciones en el campo de las bases de datos supuso un importante paso en la investigación de los SGBD, suministrando un sólido fundamento teórico para el desarrollo, dentro de este enfoque relacional, de nuevos productos.

El documento de Codd propone un modelo de datos basado en la teoría de las relaciones, en donde los datos se estructuran lógicamente en forma de relaciones "*tablas*", siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras características de tipo físico.

El trabajo publicado por Codd en 1970 presentaba un nuevo modelo de datos que perseguía una serie de objetivos, los cuales son los siguientes:

Independencia física: El modo en el que se almacenan los datos no influye en su manipulación lógica y por tanto, los usuarios que acceden a esos datos no tienen que modificar sus programas por cambios en el almacenamiento físico.

Independencia lógica: El añadir, eliminar o modificar objetos de la base de datos no repercute en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).

Flexibilidad: En el sentido de poder presentar a cada usuario los datos de la forma en que éste prefiera.

Uniformidad: Las estructuras lógicas de los datos presentan un aspecto uniforme, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.

Sencillez: Las características anteriores, así como unos lenguajes de usuario muy sencillos, producen como resultado que el modelo de datos relacional sea fácil de comprender y de utilizar por parte del usuario final.

El modelo Relacional se divide en 3 partes: estructura de los datos, integridad de los datos, y manipulación de los datos.

Estructura del Modelo Relacional

La relación es el elemento básico del modelo relacional y se representa por una tabla. Informalmente, los términos y sus equivalentes son:

Relación	Tabla
Tupla	Fila
Atributo	Columna
Número de tuplas	Cardinalidad
Número de atributos	Grado
Dominio	Colección de valores
Clave primaria	Identificador único para la tabla

Tabla 1.1.- Estructura del modelo relacional.

Es importante señalar que la tabla es plana en el sentido de que el cruce de una fila y una columna sólo puede dar un valor, es decir, no se admiten atributos multivaluados.[6]

Los principales componentes del modelo relacional son:

Relación

Como ya se menciono la relación es el elemento fundamental del modelo relacional, y básicamente se puede apreciar como una tabla, que consta de filas y columnas.

Tupla

Una tupla o fila es un registro en el cual se almacenan los datos por ejemplo de un empleado.

Atributo

Un atributo es aquel que participa en la descripción de las entidades y que como tal constituye una pieza específica de información para un determinado dominio.

En el caso de que sean varios los atributos de una misma tabla, definidos por el mismo dominio, habrá que darles nombres distintos, ya que una tabla no puede tener dos atributos con el mismo nombre, sin embargo, existen los dominios compuestos los cuales están conformados por otros dominios, además de tener un nombre y permitir aplicar restricciones.

[6] Un atributo multivaluado es aquel que puede contener más de un valor.

Dominio

Un Dominio es un conjunto finito de valores *homogéneos* y *atómicos* caracterizados por un nombre. *Homogéneo* significa que los valores son todos del mismo tipo y *atómicos* significa que son indivisibles, es decir, si se descomponen se perdería la semántica del dominio

Todo dominio tiene un nombre y un tipo de datos. La importancia de los dominios es que restringen las comparaciones, es decir, sólo se pueden comparar atributos definidos sobre el mismo dominio.

Claves

El término *clave* en una relación es un conjunto no vacío de atributos que identifican a cada tupla para que no existan datos repetidos. Toda relación siempre tendrá una clave candidata, estas claves pueden ser clasificadas en tres principales grupos:

Clave primaria: Es aquella clave que permite identificar las tuplas de la relación de forma única.

Clave alternativas o candidatas: Son aquellas claves que no han sido escogidas como claves primarias, pero que también podrían identificar de manera única a una tupla.

Clave foránea: Es un atributo en una tabla que hace referencia a la clave primaria de otra tabla, esta nos sirve para realizar las relaciones entre tablas.

Restricciones

Las restricciones son estructuras no permitidas. Hay dos tipos: inherentes y del usuario. Las inherentes son aquellas propias al modelo, por ejemplo, no tener tuplas repetidas y las del usuario son aquellas que validan las instancias de la relaciones.

Reglas de Codd.[7]

En la década de los 80's comenzaron a aparecer numerosos SGBD que se anunciaban como relacionales; sin embargo estos sistemas carecían de muchas características que se consideraban importantes en un sistema relacional, perdiendo muchas ventajas del modelo relacional. Como ejemplo externo de esto "sistemas relacionales" eran simplemente sistemas que utilizaban tablas para almacenar la información, no disponiendo de elementos como claves primarias, etc. En 1984 Codd publicó 12 reglas que un verdadero sistema relacional debería cumplir.

0. Cualquier BDMS que proclame ser relacional, deberá manejar, completamente, las bases de datos por medio de sus capacidades relacionales.

1. Regla de información. Toda la información dentro de una base de datos relacional se representa de manera explícita a nivel lógico y exactamente de una sola manera, como valores en una tabla.

2. Regla del acceso garantizado. Se garantiza que todos y cada uno de los datos (valor atómico) en una base de datos relacional pueden ser leídos recurriendo a una combinación del nombre de la tabla, valor de la llave primaria y nombre de la columna.

[7] Codd fue un matemático que invento el modelo relacional, además de inventar las reglas de codd para saber cuando un manejador de base de datos es realmente relacional.

3. El manejo sistemático de los valores nulos. En un DBMS totalmente relacional se soportan los valores nulos (que son distintos de una cadena de caracteres vacía o de una cadena con caracteres en blanco o de cero o cualquier otro número), para representar información faltante o no aplicable de una forma consistente, independientemente del tipo de dato.

4. Catálogo dinámico en línea basado en un modelo relacional. La descripción de la base de datos se representa en el nivel lógico de la misma forma que los datos ordinarios, de tal suerte que los usuarios autorizados puedan aplicar el mismo lenguaje relacional para consultarla, que aquél que emplean para con sus datos habituales.

5. Regla del sub-lenguaje de dato completo. Se debe contar con un sub-lenguaje que contemple la definición de datos, la definición de vistas, la manipulación de datos, las restricciones de integridad, la autorización, el inicio y fin de una transacción.

6. Regla de actualización de vistas[8]. Todas las vistas que teóricamente sean actualizables deberán ser actualizadas por medio del sistema.

7. Inserción, actualización y eliminación de alto nivel. La posibilidad de manejar una relación base o una relación derivada como un sólo operador se aplica a la lectura, inserción, modificación y eliminación de datos.

8. Independencia física de los datos. Los programas de aplicación y la actividad en terminales no deberán ser afectados por cambios en el almacenamiento físico de los datos o en el método de acceso.

9. Independencia lógica de los datos. Los programas de aplicación y la actividad en terminales no deberán ser afectados por cambios de cualquier tipo que preserven la información y que teóricamente permitan la no afectación en las tablas base.

10. Independencia de la integridad. Las restricciones de integridad de una base de datos deberán poder definirse en el mismo sub-lenguaje de datos relacional y deberán almacenarse en el catálogo, no en los programas de aplicación.

11. Independencia de la distribución. Un DBMS relacional tiene independencia de distribución.

12. Regla de la no subversión. Si un sistema relacional tiene un lenguaje de bajo nivel (un sólo registro cada vez), ese bajo nivel no puede ser utilizado para suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

Bases de Datos relacionales.

Se dice que una base de datos no es más que un conjunto de información relacionada, que esta agrupada o estructurada.

El archivo por sí mismo, no constituye una base de datos, sino más bien la forma en que está organizada esta información es la que da origen a la base de datos. Las bases de datos manuales, pueden ser difíciles de gestionar y modificar, por ejemplo, en una guía de teléfonos no es posible encontrar el número de un individuo si no sabemos su apellido, aunque conozcamos su domicilio.

[8] Una vista es una tabla virtual que se crea para mostrar datos que se requieren con mucha frecuencia o para que algunas personas que no tengan permiso de ver ciertos campos solo vean lo que tienen permitido ver.

Desde el punto de vista informático, una base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulan ese conjunto de datos.

Desde el punto de vista más formal, se puede definir una base de datos como un conjunto de datos estructurados, fiables y homogéneos, organizados independientemente en una máquina, accesibles a tiempo real, compartidas por usuarios concurrentes que necesitan información diferente y no predecible en el tiempo.

La idea general es que estamos tratando con una colección de datos que cumplen las siguientes propiedades:

- Son independientes de las aplicaciones y del soporte de almacenamiento.
- Presentan la menor redundancia posible.
- Son compartidos por varios usuarios y/o aplicaciones.

Los sistemas relacionales son importantes porque ofrecen muchos tipos de procesos de datos, como: simplicidad y generalidad, facilidad de uso para el usuario final, períodos cortos de aprendizaje y las consultas de información se especifican de forma sencilla. Las tablas son un medio para representar la información de una forma más compacta y al mismo tiempo acceder a información contenida en dos o más tablas.

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas.

Diseño de las bases de datos relacionales

El primer paso para crear una base de datos, es planificar el tipo de información que se quiere almacenar en la misma, teniendo en cuenta dos aspectos: la información disponible y la información que necesitamos.

El diseño de la estructura de una tabla consiste en una descripción de cada uno de los campos que componen el registro y los valores o datos que contendrá cada uno de esos campos.

Los campos son los distintos tipos de datos que componen la tabla, por ejemplo: nombre, apellido, domicilio. La definición que un campo requiere principalmente es el nombre del campo, el tipo de campo y la longitud del mismo.

Los registros constituyen la información que va contenida en los campos de la tabla, por ejemplo, en la base de datos para un hospital sería el nombre del paciente, el apellido del paciente y la dirección del mismo.

El lenguaje de consulta relacional.

El SQL[9] es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. El nombre "SQL" es una abreviatura de Structured Query Languaje (Lenguaje de consultas estructurado). Como su propio nombre indica, SQL es un lenguaje informático que se puede utilizar para interaccionar con una base de datos y más concretamente con un tipo especifico llamado base de datos relacional.

El algebra relacional consiste en una colección de operaciones sobre relaciones donde cada operación toma una o más relaciones como su operando y produce otra relación como su resultado. Dado que el resultado de

[9] SQL es el lenguaje estándar que utilizan los manejadores de base de datos relacionales para poder desempeñar todas sus capacidades, este lenguaje fue propuesto por Codd.

una operación del álgebra relacional es una relación, ésta a su vez puede ser sujeto de posteriores operaciones algebraicas.

El algebra relacional se basa en la teoría de conjuntos, relaciones y en el álgebra de conjuntos. Adicionalmente al conjunto básico de operadores como: unión, diferencia, producto cartesiano e intersección; incorpora operadores específicos de base de datos tales como proyección, selección y join.

Unión

Teniendo dos conjuntos, la unión representaría los elementos que se encuentran en ambos conjuntos, por ejemplo, la siguiente operación da una idea de lo antes mencionado:

$$\{1,4,5,10\} \cup \{1,4,3,9\} = \{1,3,4,5,9,10\}$$

En términos de tablas, hay que considerar que la unión sea compatible, es decir, el número de atributos debe ser el mismo y del mismo tipo de datos.

CLAVE	RFC	NOMBRE	DEPTO	SUELDO
125	LOPA650303	PEDRO LOPEZ	1	$ 250.00
236	PERM640506	MARIA PEREZ	1	$350.00

CLAVE	RFC	NOMBRE	DEPTO	SUELDO
741	SUOP701213	PABLO SUAREZ	2	$ 500.00
254	TEMM680409	MONICA TELLEZ	3	$400.00

Haciendo la unión de estas dos tablas anteriores obtendríamos:

CLAVE	RFC	NOMBRE	DEPTO	SUELDO
125	LOPA650303	PEDRO LOPEZ	1	$ 250.00
236	PERM640506	MARIA PEREZ	1	$350.00
741	SUOP701213	PABLO SUAREZ	2	$ 500.00
254	TEMM680409	MONICA TELLEZ	3	$400.00

Diferencia

Teniendo dos conjuntos, la diferencia representaría los elementos que están en uno de los conjuntos, pero que no están en el otro, por ejemplo, viendo la siguiente operación:

$$\{1,4,5,10\} - \{1,4,3,9\} = \{5,10\}$$

Producto Cartesiano

El producto cartesiano es el producto cruz entre 2 tablas, es decir, el resultado es la unión de cada renglón de una tabla con cada renglón de la otra tabla. El producto es una yuxtaposición donde los elementos son combinados o concatenados, por ejemplo:

$$\{1,4\} \times \{1,4\} = \{1,4,4,16\}$$

SOCIO

CODIGO	NOMBRE	DIRECCION
1	Elena	D.F.
2	Manuel	Estado de México

LIBRO

ID_LIBRO	LIBRO	AUTOR	EDITORIAL
4	BD	Gardarin	McGraw Hill
5	INFORMIX	Zeroual	Ra-Ma

SOCIO X LIBRO

CODIGO	NOMBRE	DIRECCION	ID_LIBRO	LIBRO	AUTOR	EDITORIAL
1	Elena	D.F.	4	BD	Gardarin	McGraw Hill
1	Elena	D.F.	5	INFORMIX	Zeroual	Ra-Ma
2	Manuel	Estado de México	4	BD	Gardarin	McGraw Hill
2	Manuel	Estado de México	5	INFORMIX	Zeroual	Ra-Ma

Intersección

Teniendo dos conjuntos, la intersección representaría los elementos que están en uno de los conjuntos y que además existen en el otro, por ejemplo, la siguiente operación:

$$\{1,4,5,10\} \cap \{1,4,3,9\} = \{1,4\}$$

Proyección

La proyección selecciona y genera un subconjunto con los atributos indicados de una tabla. También es conocida como operación vertical.

EMPLEADO

CLAVE	RFC	NOMBRE	DEPTO	SUELDO
125	LOPA650303	PEDRO LOPEZ	1	$ 250.00
236	PERM640506	MARIA PEREZ	1	$350.00
741	SUOP701213	PABLO SUAREZ	2	$ 500.00
254	TEMM680409	MONICA TELLEZ	3	$400.00

Al aplicar la operación de proyección se haría lo siguiente:

```
SELECT CLAVE,RFC,NOMBRE
FROM EMPLEADO;
```

Entonces esta consulta nos daría como resultado:

EMPLEADO

CLAVE	RFC	NOMBRE
125	LOPA650303	PEDRO LOPEZ
236	PERM640506	MARIA PEREZ
741	SUOP701213	PABLO SUAREZ
254	TEMM680409	MONICA TELLEZ

Selección

La selección toma y genera un subconjunto con los renglones indicados de una tabla. También es conocida como operación horizontal.

EMPLEADO

CLAVE	RFC	NOMBRE	DEPTO	SUELDO
125	LOPA650303	PEDRO LOPEZ	1	$ 250.00
236	PERM640506	MARIA PEREZ	1	$350.00
741	SUOP701213	PABLO SUAREZ	2	$ 500.00
254	TEMM680409	MONICA TELLEZ	3	$400.00

Ahora aplicando la operación de selección se obtendría el siguiente resultado.

```
SELECT *
FROM EMPLEADO
WHERE CLAVE IN (125,741);
```

EMPLEADO

CLAVE	RFC	NOMBRE	DEPTO	SUELDO
125	LOPA650303	PEDRO LOPEZ	1	$ 250.00
741	SUOP701213	PABLO SUAREZ	2	$ 500.00

Join

La operación join es en esencia un producto cartesiano, donde se seleccionan las columnas que satisfagan las condiciones indicadas. Es la operación más común en las bases de datos relacionales.

1.1.10 La independencia de datos y la integridad referencial.

La independencia de los datos

Una de las principales ventajas que provee una base de datos es la independencia entre los datos y los tratamientos que se hacen de ellos ya que en los sistemas orientados a procesos los datos eran sumamente dependientes de los programas.

El concepto de base de datos rescata aquella dependencia que tienen los procesos de los datos y la radicaliza priorizando la independencia de estos últimos, determinando mecanismos de definición y de descripción que no requieren de procesos.

Como tal, la independencia de los datos se refiere a la protección contra los programas de aplicación que puedan originar modificaciones cuando se altera la organización física o lógica de la base de datos.

Existen 2 niveles de independencia de datos:

- *Independencia física de datos:* Es la capacidad de modificar el esquema físico sin provocar que se vuelvan a escribir los programas de aplicación.

- *Independencia lógica de datos:* Capacidad de modificar el esquema conceptual sin provocar que se vuelvan a escribir los programas de aplicación.

La integridad referencial

El término de integridad referencial se enmarca en la segunda regla de integridad y se aplica a las claves foráneas:

"Si en una relación hay alguna clave foránea, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos".

Lo que en realidad trata de decir el texto anterior es que las claves foráneas no pueden dejar de tener correspondencia con la clave primaria de la tabla externa; las tuplas que contienen claves foráneas que no tienen una clave candidata, se denominan entidades huérfanas. Existen además otros tipos de claves:

Clave primaria: Es aquel atributo que identifica de manera única a un registro. Esto es, no debe haber dos tuplas que tengan el mismo valor, por lo tanto, con sólo conocer el valor de la clave primaria para una determinada tupla será suficiente para identificarlo de manera única.

Clave candidata: Es el atributo o conjunto de atributos que podrían servir como llaves primarias.

Clave secundaria: Son aquellas claves candidatas que no se eligieron como llave primaria, es decir, tienen todas las características para ser claves primarias, pero que por alguna razón no fueron tomadas como tal debido quizás a que hubo otra que cumplía mejor con ese objetivo.

Clave foránea: Es una clave primaria en otra relación, estas representan las asociaciones entre las diferentes entidades, es decir, son claves que están siendo compartidas por dos tablas para formar una relación entre ellas.

Además la integridad referencial sirve para determinar las consecuencias que pueden tener ciertas operaciones (borrado y modificación) realizadas sobre tuplas de la relación; pudiéndose distinguir estas restricciones:

Operación restringida: Esto es, el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada; sólo se permite si no existen tuplas con dicha clave en la relación que contiene la clave foránea. Esto llevaría, por ejemplo, a que para poder borrar una editorial de la base de datos no tendría que haber ningún libro que estuviese publicado por dicha editorial, en caso contrario el sistema impediría el borrado.

Operación con transmisión en cascada: El borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo el borrado o modificación en cascada de las tuplas de la relación que contienen la clave foránea. En el ejemplo, equivaldría a decir que al modificar el nombre de una editorial en la relación EDITORIAL, se tendría que modificar también dicho nombre en todos los libros de la base de datos publicados por dicha editorial.

Operación con puesta a nulos: En el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo poner a nulos los valores de las claves foráneas de la relación que referencia. Esto llevaría a que cuando se borra una editorial, a los libros que ha publicado dicha editorial y que se encuentran en la relación LIBROS se les coloque el atributo *nombre_editorial* a nulos. Esta opción sólo es posible cuando el atributo que es clave foránea admite el valor nulo.

Operación con puesta a valor por defecto: El borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada llevan consigo poner el valor por defecto a la clave foránea de la relación que referencia.

1.1.11 Modelo Entidad-Relación.

El modelo entidad-relación (E-R) es el modelo conceptual más utilizado para el diseño de bases de datos. Fue introducido por Peter Chen[10] en 1976. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Por lo tanto se puede decir que el modelo entidad - relación, es una técnica de diseño de bases de datos gráfica, que incorpora información relativa a los datos y la relación existente entre ellos, para poder así plasmar una visión del mundo real sobre un soporte informático y que se caracteriza fundamentalmente por:

- Sólo reflejar la existencia de los datos sin expresar lo que se hace con ellos.
- La independencia de la base de datos y de los sistemas operativos.
- La inclusión de todos los datos sin considerar las aplicaciones que se tendrán.

En esencia el modelo E-R, consiste en buscar las entidades que describan los objetos que intervienen en el problema y las relaciones entre esas entidades.

Esto ayuda al programador durante la codificación y por otro lado, al usuario a comprender el problema y el funcionamiento del programa.

Para comprender un poco más este diagrama se debe definir algunos conceptos importantes en este modelo de datos.

Entidades

Se puede definir como entidad a cualquier objeto, real o abstracto, que existe en un contexto determinado o que puede llegar a existir y del cual se desea guardar información, por ejemplo, un profesor, un alumno o bien una materia. Las entidades se pueden clasificar en:

- *Regulares:* Son aquellas entidades que existen por sí mismas, es decir, la existencia de un ejemplar de la entidad no depende de la existencia de otros ejemplares en otra entidad, por ejemplo, la entidad "PROFESOR".

- *Débiles:* Son aquellas entidades en las que su existencia depende de la existencia de ejemplares en otras entidades, por ejemplo, la existencia de la entidad “PROFESOR” depende de la existencia de la entidad “ESCUELA”.

Las entidades se representan gráficamente por medio de un rectángulo, y en su interior contiene el nombre de la entidad como se muestra en la fig. 1.6.

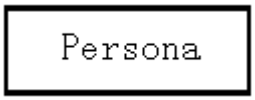


Fig. 1.6.- Representación de una entidad.

[10] Peter Chen fue un doctor en ciencias de la computación y matemáticas aplicadas, además de ser el inventor del modelo entidad-relación en 1976.

Atributos

Las entidades se componen de atributos que son cada una de las propiedades o características que tienen las entidades. Cada ejemplar de una misma entidad posee los mismos atributos, tanto en nombre como en número, diferenciándose cada uno de los ejemplares por los valores que toman dichos atributos.

Si se considera la entidad "PROFESOR" y se definen sus atributos, estos serian: Nombre, Cursos, Teléfonos y Edad, y otros más.

Existen cuatro tipos de atributos:

- *Obligatorios:* Aquellos que forzosamente deben tomar un valor.
- *Opcional:* Aquellos atributos que pueden o no tener valores.
- *Monoevaluado:* Aquel atributo que sólo puede tener un único valor.
- *Multievaluado:* Aquellos atributos que pueden tener varios valores.

También existe otro tipo de atributo, el cual es llamado derivado, ya que nace a partir de otro atributo, un ejemplo de esto es cuando se tiene la fecha de nacimiento de un alumno y este atributo permite formar otro atributo más el cual puede ser la edad, este atributo se representa al igual que los atributos normales con la diferencia de que su representación grafica es punteada.

Los atributos se representan gráficamente por medio de una elipse, y en su interior contiene el nombre del atributo como se aprecia en la fig. 1.7.

Fig. 1.7.-Representación de un atributo.

En determinadas ocasiones es indispensable la descomposición de un atributo para definir su dominio, por ejemplo un teléfono celular contiene un prefijo antes del número.

Dominios

Un dominio se define como un conjunto de valores que puede tomar un determinado atributo dentro de una entidad.

Su representación gráfica se aprecia por medio de un hexágono, el cual describe el dominio de los atributos como se muestra en la fig. 1.8.

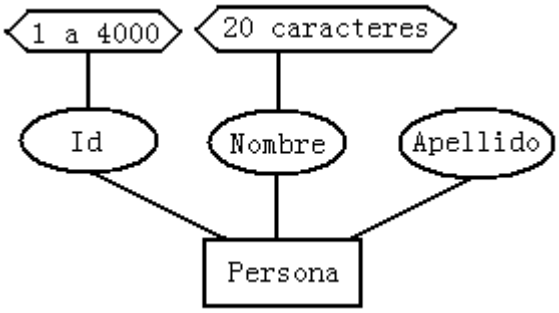


Fig. 1.8.-Representación de un dominio de un atributo.

Claves

El modelo Entidad - Relación exige que cada entidad tenga un identificador, se trata de un atributo o conjunto de atributos que identifican de forma única a cada uno de los ejemplares de la entidad.

Estos identificadores reciben el nombre de Clave Primaria o Primary Key (PK).[11] Puede ser que existan más identificadores que pueden ser claves primarias, pero por alguna u otra circunstancia no lo son, a estos atributos se les conoce como Identificadores Candidatos (IC).

Para poder identificar un atributo que es clave primaria de la entidad se subraya el nombre del atributo como se observa en la fig. 1.9.

Fig. 1.9.- Atributo clave.

Interrelaciones

Se entiende por interrelación a la asociación, vinculación o correspondencia entre entidades. Por ejemplo, entre la entidad "PROFESOR" y la entidad "CURSO" se puede establecer la relación "*IMPARTE*" por que el profesor imparte cursos.

Al igual que las entidades, las interrelaciones se pueden clasificar en regulares y débiles, esto de acuerdo al tipo de entidad que estén asociando, entidades regulares o entidades débiles, con otra de cualquier tipo. Las interrelaciones débiles se subdividen en dos grupos:

- *En existencia*: Cuando los ejemplares de la entidad débil no pueden existir si desaparece el ejemplar de la entidad regular del cual dependen.
- *En identificación*: Cuando además de ser una relación en existencia, los ejemplares de la entidad débil no se pueden identificar por sí mismos y exigen añadir el identificador principal de la entidad regular del cual dependen para ser identificados.

La representación gráfica de una interrelación es un rombo como el que se muestra en la fig. 1.10, en el cual en su interior la palabra que relaciona a las dos entidades.

Fig. 1.10.- Representación de una interrelación.

Las interrelaciones débiles se representan con un doble rombo.

Cardinalidad

La cardinalidad especifica el número mínimo y el máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad.

[11] Primary Key (PK) es un identificador único que nos garantiza que el atributo no puede repetirse.

Son posibles tres tipos de cardinalidades:

- Relaciones *de uno a uno*: una instancia de la entidad A se relaciona con una y solamente una de la entidad B.
- Relaciones de uno a muchos: cada instancia de la entidad A se relaciona con varias instancias de la entidad B.
- Relaciones de muchos a muchos: cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B.

El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M", aunque la notación depende del lenguaje utilizado, la que más se usa actualmente es el unificado como se muestra en la fig. 1.11.

Fig. 1.11.- Cardinalidad entre 2 entidades.

1.1.12 Normalización.

El proceso normalización consiste en agrupar a los campos de datos en un conjunto de relaciones o tablas que representan a las entidades, sus características y sus relaciones de forma adecuada. La razón de la normalización es asegurar que el modelo conceptual de la base de datos funcionará.

Esto no significa que una estructura no normalizada no funcionará, sino que puede causar algunos problemas cuando los programadores de aplicación traten de modificar la base de datos para insertar, actualizar o eliminar datos.

Las formas de normalización fueron propuestas originalmente por Codd, entre 1971 y 1972.
Posteriormente varios investigadores continuaron trabajando en esta teoría y a lo largo del tiempo han surgido varias formas de normalización que complementan y refuerzan a las enunciadas por Codd.

Las *formas normales* son una serie de restricciones que se definen sobre las estructuras relacionales para evitar anomalías al efectuar adiciones, eliminaciones o actualizaciones de tuplas. Con el fin de conseguir que una relación cumpla con una forma normal se efectúa un proceso de descomposición.

Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.

La normalización se lleva a cabo en una serie pasos, cada paso corresponde a una forma normal que tiene ciertas propiedades. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto y más fuerte y por lo tanto, son menos vulnerables a las anomalías de actualización.

Las tres formas normales son las siguientes:

Primera Forma Normal (1FN): Una relación está en primera forma normal, si y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna.

Segunda Forma Normal (2FN): Una relación está en segunda forma normal, si y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.

Tercera Forma Normal (3FN): Una relación está en tercera forma normal, si y sólo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. La dependencia es transitiva si existen las dependencias siendo atributos o conjuntos de atributos de una misma relación.

Básicamente, las reglas de normalización están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

1.1.13 Herramientas CASE.

Desde el inicio de la creación de software ha existido la necesidad de crear herramientas automatizadas que permitan incrementar la productividad de los diseñadores de software, en un inicio, los esfuerzos se direccionaron hacia programas traductores, recopiladores, ensambladores, procesadores de macros, montadores y cargadores.

Al ver los beneficios de este conjunto de aplicaciones se genero una gran demanda por nuevo software con características similares. El significado de las siglas CASE viene de su acrónimo en ingles Computer Aided Assisted Automated Software Systems Engineering.

Ahora, las herramientas CASE se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante las diferentes etapas del ciclo de desarrollo del Software: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

Las Herramientas CASE se iniciaron con un procesador de palabras que fue usado para crear y manipular documentación. Los 70's vieron la introducción de técnicas gráficas y diagramas de flujo de datos.

La primera herramienta comercial se remonta a 1982, aunque algunos especialistas indican que algunos ejemplos de herramientas para diagramación ya existían.

No fue sino hasta 1985 cuando las herramientas CASE se volvieron realmente importantes en el proceso de desarrollo de software. Los proveedores prometieron a la Industria que muchas actividades serían beneficiadas por la ayuda de las CASE.

El objetivo en 1985 para muchos vendedores era producir software más rápidamente y las herramientas CASE fueron una familia de métodos que favorecieron la planeación, el análisis y el diseño además de que también brindaron beneficios para la mejora en la calidad, la fiabilidad, la utilidad y el rendimiento.

Clasificación de las Herramientas Case

Las herramientas CASE, en función de las fases del ciclo de vida que cubre, se pueden agrupar de la forma siguiente:

1. Herramientas integradas, I-CASE (Integrated CASE): Abarcan todas las fases del ciclo de vida del desarrollo de sistemas son llamadas CASE workbench.

2. Herramientas de alto nivel, U-CASE (Upper CASE): Orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo, análisis y diseño.

3. Herramientas de bajo nivel, L-CASE (Lower CASE): Dirigidas a las últimas fases del desarrollo, construcción e implantación.

4. Juegos de herramientas, (Tools CASE): Son el tipo más simple de Herramientas CASE, automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

Herramientas Case más utilizadas

ERwin: Es una herramienta para el diseño de base de datos, que Brinda productividad en su diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada, además ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de stored procedure y triggers para los principales tipos de base de datos.

EasyCASE: Es un producto para la generación de esquemas de base de datos e ingeniería reversa, esta herramienta permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación, para poder crear las aplicaciones eficazmente desde el procesamiento de transacciones a la aplicación de bases de datos de cliente/servidor, así como sistemas de tiempo real, es una herramienta multi-usuario.

Oracle Designer: Oracle Designer es un conjunto de herramientas para guardar las definiciones que necesita el usuario y automatizar la construcción rápida de aplicaciones cliente/servidor gráficas. Este esta integrado con Oracle Developer, lo cual provee una solución para desarrollar sistemas empresariales de segunda generación. Todos los datos ingresados por cualquier herramienta de Oracle Designer, en cualquier fase de desarrollo, se guardan en un repositorio central.

DBDesigner: Producto destacable por su sencillez, permite modelar sobre MySQL y dispone de la capacidad de generar documentación e incluso pantallas de administración sobre PHP.

TableDesigner: Herramienta que permite la creación de bases de datos en Access y SQL Server de Microsoft.

1.2.-SISTEMAS MANEJADORES DE BASES DE DATOS RELACIONALES (RDBMS).

1.2.1 ¿Qué es un RDBMS?

Entre la base de datos física (es decir, los datos tal y como están almacenados en la realidad) y los usuarios del sistema, existe un nivel de programas, denominado, sistema manejador de bases de datos (SMBD) o, en la mayoría de los casos, el sistema administrador de bases de datos DBMS (Data Base Management System).

En forma general se podía decir que es un software que con la ayuda del sistema operativo donde se encuentra instalado ayuda a administrar las bases de datos relacionales.

Un RDBMS es el conjunto de programas que permiten la definición, manipulación y control de acceso a los datos, y las relaciones que existen entre ellos.

Algunas características de los RDBMS son:

- Facilitan la integridad, seguridad y acceso de los datos.
- Los datos se almacenan como mínima redundancia.
- Las aplicaciones son independientes del almacenamiento físico de los datos.

1.2.2 Función del RDBMS.

Un RDBMS debe permitir las siguientes condiciones en una base de datos:

- Los datos han de estar almacenados juntos, es decir, que aunque estén en diferente partición de disco o en diferentes discos el RDBMS dan la impresión que los datos están almacenados juntos.

- Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento.

- Los datos son compartidos por diferentes usuarios y programas de aplicación; existe un mecanismo común para la inserción, actualización, borrado y consulta de los datos.

- Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos.

- Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes RDBMS.

- Proporciona seguridad en los datos, esto se refiere a que otorga y restringe permisos a los usuarios, ya sea para su conexión o en el manejo de los objetos de las bases de datos o en el manejo propio de los datos.

Conceptualmente lo que sucede en un RDBMS cuando un usuario realiza alguna petición, se presenta lo siguiente:

1. El usuario solicita alguna petición a la base de datos empleando algún sublenguaje de datos determinado (SQL).

2. El RDBMS interpreta esa solicitud y la analiza.

3. El RDBMS inspecciona la sintaxis de la petición y si es correcta verifica si el usuario tiene los privilegios de acceder al RDBMS, y una vez que se cerciora que es un usuario autorizado ahora verifica que tenga privilegios sobre los objetos de la base de datos, como lo son las vistas, tablas, etc.

4. El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada y devuelve una respuesta al usuario.

El RDBMS cuenta con un esquema de seguridad de tres niveles, esto es un acceso multicapas como se muestra en la fig. 1.12, el cual ayuda a tratar de preservar los datos de usuarios sin privilegios, que no tengan acceso a ellos.

- El usuario final debe tener una cuenta válida dentro de la capa del servidor (DBMS). *Seguridad a nivel servidor*.

- El usuario final debe ser un usuario válido dentro de la capa de la base de datos. *Seguridad a nivel base de datos*.

- El usuario final deberá tener permisos dentro de la capa de los datos. *Seguridad a nivel de permisos sobre objetos y comandos*.

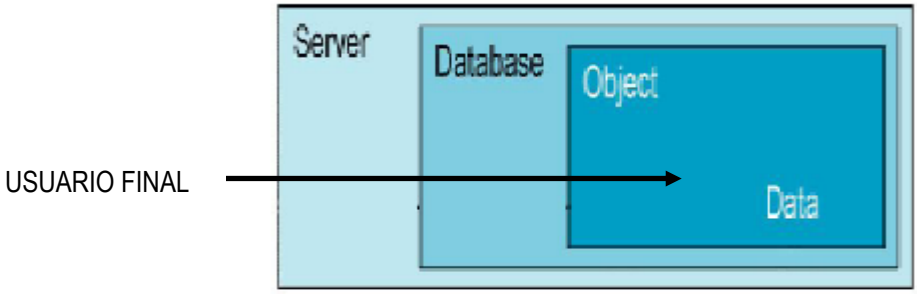


Fig. 1.12.- Esquema de seguridad de 3 niveles.

1.2.3 Componentes de los RDBMS.

Los componentes principales del RDBMS son: DDL, DML, DCL y el DD, los cuales a continuación se describen cada uno de ellos.

DDL (Lenguaje de Definición de Datos).

Se utiliza para crear, eliminar o modificar tablas, índices, vistas, triggers, procedimientos; es decir, permite definir la estructura de la base de datos mediante comandos como crear (*CREATE*), eliminar (*DROP*), o alterar (*ALTER*). Este componente actúa directamente sobre los objetos de las base de datos.

-*CREATE.* Utilizado para crear nuevas bases de datos, tablas, campos, índices, vistas, defaults, reglas, procedimientos, procedimientos, triggers.

-*ALTER.* Utilizado para modificar la estructura de una tabla para agregar campos o constrainst.

-*DROP.* Utilizado para eliminar bases de datos, tablas, campos, índices, vistas, defaults, reglas, procedimientos, procedimientos, triggers.

DML (Lenguaje de Manipulación de Datos).

Se utiliza para realizar la consulta y edición de la información contenida en la base de datos, esto implica: seleccionar, insertar, borrar, modificar. Este componente actúa sobre los datos que se encuentran en la base de datos.

Las instrucciones relacionadas con este componente son:

-*SELECT.* Permite realizar consultas a la base de datos, es decir, seleccionando los datos que queramos conocer.

-*INSERT.* Empleado para agregar registros a una tabla.

-*UPDATE.* Utilizado para modificar los valores de los campos de una tabla.

-*DELETE.* Utilizado para borrar los valores de los campos de una tabla.

DCL (Lenguaje de Control de Datos).

Se utiliza para la definición de los privilegios de control de acceso y edición a los elementos que componen la base de datos (seguridad), es decir, permitir o revocar el acceso. Actúa directamente sobre los permisos de la base de datos.

Los permisos a nivel base de datos pueden otorgarse a usuarios para ejecutar ciertos comandos dentro de la base o para que puedan manipular objetos y los datos que puedan contener éstos.

Las instrucciones relacionadas con este componente son:

-*GRANT.* Permite otorgar permisos a los usuarios sobre los objetos definidos en la base de datos, así como las operaciones a utilizar sobre ellos.

-*REVOKE.* Permite revocar permisos sobre los objetos definidos en la base de datos y las operaciones sobre los mismos.

DD (Diccionario de Datos).

El contenido del diccionario puede considerarse como "datos acerca de los datos" (los cuales comúnmente reciben el nombre de metadatos[12]), es decir, definiciones de otros objetos de la base de datos.

En particular, todos los diversos esquemas (externo, conceptual e interno), se almacenan físicamente en el diccionario, tanto en forma fuente como en forma objeto. Un diccionario amplio incluirá también las referencias cruzadas que indican, por ejemplo que partes de datos utiliza cada programa, que informes necesita cada departamento, etc. De hecho, el diccionario puede integrarse a la base de datos que describe, y, por tanto, incluir su propia descripción.

Debe ser posible consultar el diccionario de la misma manera que cualquier otra base de datos, de modo que, por ejemplo, el DBA (Administrador de la Base de Datos) pueda describir con facilidad que programas tienen probabilidad de ser afectados por un cambio propuesto al sistema.

Básicamente el diccionario de datos lo que realiza es la descripción de todos los elementos del sistema. El propio RDBMS cuenta con su DD que describe sus elementos.

1.2.4 SQL ANSI 89, 92 y 99.

El SQL ANSI es un estándar que tratan de seguir los RDBMS que existen en el mercado, y éste se hizo debido a que cada RDBMS utilizaba sus propia administración de las bases de datos y para que éstos fueran de cierta forma compatibles.

[12] Un metadato es una descripción de los datos, se puede decir que son datos acerca de los datos.

Existen 3 versiones, las cuales son el SQL ANSI 89, 92 y 99, este último es el que esta vigente como estándar para los RDBMS, y algunas de sus características se pueden apreciar a continuación.

Características principales del SQL ANSI 89.

I. Agregan la capacidad conocida como integridad referencial y la descripción de todo el modelo relacional.

II. Se definió que el lenguaje SQL está compuesto por comando, cláusulas, operadores y funciones de agregado[13]. Estos elementos se combinan para definir y manipular la base de datos.

III. Se establecen los elementos de un DBMS (DDL, DML y DCL), así como las instrucciones y sintaxis relacionadas con cada uno de ellos.

IV. Establecimiento de las cláusulas del comando SELECT, las cuales son: *FROM, WHERE, GROUP BY, HAVING, ORDER BY*.

V. Definición de los operadores lógicos: AND, OR y NOT.

VI. Definición de los operadores de comparación.

VII. Se determinan las funciones de agregado, tales como: AVG, COUNT, SUM, MAX, MIN.

Características principales del SQL ANSI 92.

I. Toma todas características definidas en el estándar SQL ANSI 89.

II. Permite la definición de esquemas.

III. Permite la definición de dominios por parte de los usuarios, es decir, tipos de datos definidos por el usuario.

IV. Menciona las consideraciones para realizar consultas sencillas, multi-tablas y subconsultas.

V. Incluye los operadores EXISTS y NOT EXISTS.

VI. Contempla el uso de la palabra DISTINCT en una consulta.

VII. Menciona algunas consideraciones para el uso de las cláusulas GROUP BY y HAVING.

VIII. Especifica la definición de vistas en una base de datos.

Características principales del SQL ANSI 99.

I. Toma todas características definidas en los estándares SQL ANSI 89 y 92.
II. Incluye nuevos tipos de datos escalares: BOOLEAN, CLOB (objeto de caracteres largo) y BLOB (objeto binario grande).

[13] Las funciones de agregado son funciones predefinidas que nos ayudan a realizar ciertas cosas de una manera más fácil.

III. Presenta dos nuevos operadores de totales: EVER y ANY.

IV. Incorpora generadores de tipo de dato: REF, ARRAY y ROW.

V. Soporta una opción LIKE en CREATE TABLE, lo cual permite que todas o algunas definiciones de columna de una nueva tabla sean copiadas a partir de otra ya existente.

VI. Incluye la cláusula WITH para introducir nombres abreviados para determinadas expresiones.

VII. Incorpora una nueva expresión de condición IS DISTINCT para la cláusula FROM.

1.2.5 Principales RDBMS comerciales y sus características.

Los principales RDBMS más utilizados que existen en el mercado son los siguientes:

Microsoft SQL Server

Las características para la instalación se muestran en la tabla 1.2.

Sistema Operativo	Windows NT Server Windows 2000 Server Windows 2000 Advanced Server Windows 2000 Data Center Windows Server 2003
Hardware	PIII o superior, según los requerimientos del sistema operativo.
Memoria	Enterprise Edition: 64 MB como mínimo. Standard Edition: 32 MB como mínimo.
Disco Duro	SQL Server 2000, instalación completa 180 MB SQL Server 2000, instalación típica 170 MB SQL Server 2000, Instalación mínima 65 MB SQL Server 2000, solo herramientas cliente 90 MB
Observaciones	Es necesario disponer de Microsoft Windows 2000 Server para algunas características de SQL Server 2000. Nota: los requerimientos de hardware y software cambian dependiendo de la versión que se desee instalar.

Tabla 1.2.- Caracteristicas de instalación de SQL Server.

Características principales:

- Compatibilidad con estándares de W3C, incluyendo XML, Xpath, XSL, HTTP.
- Obtiene código XML de las consultas realizadas con SQL.
- Manipulación de documentos XML.
- Manejo de bases de datos distribuidas[14].
- Manejo de varias particiones físicas para almacenamientos de datos flexibles.
- Permite realizar algunas tareas de mantenimiento y administración de la base de datos sin tener que darla de baja.

[14] Las bases de datos distribuidas son aquellas que dan la apariencia que la información esta junta pero en realidad esta distribuida en discos o máquinas diferentes.

- Permite realizar acciones OLAP (Online Analazing Processing), herramienta que permiten analizar datos almacenados en una Base da Datos, por medio de cubos de información.
- Consulta y modificación de cubos virtuales de manera gráfica.
- Conectividad con clientes ODBC[15] y JDBC[16].

Sybase

Las características para la instalación se muestran en la tabla 1.3.

<table>
<tr><td rowspan="3">Plataformas Soportadas</td><td colspan="3">Sistema Operativo</td></tr>
<tr><td>Empresa</td><td>32 Bits</td><td>64 Bits</td></tr>
<tr><td>Sun Microsystems
Hewlett Packard
IBM
SGI
Compaq
Linux
Microsoft Windows</td><td>HP-UX
AIX
IRIX
Red Hat
NT,98 (sólo clientes)</td><td>Solaris
HP-UX
AIX
TRU64</td></tr>
<tr><td rowspan="2">Memoria</td><td colspan="2">Sistema Operativo</td><td>Memoria</td></tr>
<tr><td colspan="2">AXP
HP 32
HP 64
Linux
NT
Sun 32
Sun 64</td><td>94 MB
64 MB
90 MB
32 MB
46 MB
66MB
92 MB</td></tr>
<tr><td>Disco Duro</td><td colspan="3">240 MB</td></tr>
<tr><td>Especificaciones del servidor</td><td colspan="3">▪ Bases de datos totales 32,767
▪ Tamaño de la base datos 4 TB
▪ Bases de datos en update 16
▪ Tablas en una consulta 50
▪ Usuarios por base de datos 2,146,484,223
▪ Columnas por tabla 1024
▪ Tamaños de página 2k, 4k, 8k, 16k
▪ Argumentos por procedimiento 1024</td></tr>
</table>

Tabla 1.3.- Caracteristicas de instalación de Sybase.

Características principales:

- Diseñado para soportar aplicaciones OLTP (On Line Transaction Procesor, ambiente diseñado para insertar, actualizar y borrar datos en una base datos).
- Conectividad con clientes ODBC y JDBC.
- Soporte para BLOB's (Large Objects).
- Permite realizar queries XQL, lo cual significa que utiliza un motor abierto para búsqueda dentro de contenidos XML almacenados en la base de datos, o en un URL.

[15] ODBC es un conector que se utiliza para enlazar los programas con las bases de datos, este conector fue desarrollado por Microsoft.
[16] JDBC es un conector que al igual que ODBC nos ayuda a la conexión entre un programa con la base de datos, aunque ese conector es utilizado por los programas realizados por Java.

- Tamaño expandido de filas y datos, es decir soporta filas más grandes, columnas más grandes. Se soportan ahora tamaños de páginas de 2k, 4k, 8k o 16k (entre más tamaño de página mayor rendimiento de operaciones SQL).
- Bloqueo a: nivel de fila, nivel de página de datos, nivel de página (datos e índices), nivel de tabla.

Oracle

Características Principales:

- Ofrece varias plataformas de desarrollo para Internet y aplicaciones tradicionales, tales como: XML, Enterprise Java Engine, SQL y PL/SQL, C, C++, entre otras.
- Soporte Unicode.
- Extiende las habilidades de una base de datos para Internet.
- Amplía distintos mecanismos para protección de datos.
- Soporta OLTP y OLAP.
- Contiene mecanismos de gran funcionalidad y flexibilidad para compartir la información almacenada en la base de datos con otras bases de datos o aplicaciones.
- Conectividad con clientes ODBC y JDBC.
- Soporte para BLOB's.
- Ofrece escalabilidad y performance sin modificar las aplicaciones instaladas.
- Soporta columnas con cifrado de datos.
- Permite replicación de bases de datos (bases de datos distribuidas).
- Ofrece distintas herramientas para la administración de la base de datos.
- Redefinición de tablas en línea.
- Respaldo y recuperación en línea.

Plataformas que soporta:

- ✓ Solaris
- ✓ HP-UX
- ✓ Compaq Tru64
- ✓ AIX
- ✓ HP Alpha
- ✓ Linux
- ✓ Windows NT/2000/XP Professional

Informix y DB2.

Características principales:

- Soporta Bases de datos de más de 4 TB.
- Soporte para acceso a la base de datos vía Web.
- Provee acceso a cualquier tipo de cliente.
- Permite manejo de base de datos distribuidas.
- Capacidad de replicación de bases de datos.
- Permite realizar query's en paralelo.
- Contiene plataformas de desarrollo con SPL: (Informix Stored Procedure Language), C, Java, XML.
- Conectividad vía ODBC, JDBC, OLE/DB.
- Soporta aplicaciones para eCommerce, e inteligencia de negocios.
- Soporta OLAP y OLTP.

Plataformas que soporta:

- ✓ IBM AIX.
- ✓ SGI IRIX.
- ✓ Sun Solaris.
- ✓ HP-UX.
- ✓ Compaq Tru64.
- ✓ Linux.
- ✓ Windows NT/2000/XP/2003.

PostgreSQL

Características principales:

- Base de datos de distribución libre.
- Velocidad.
- Confiabilidad.
- Flexibilidad.
- Bajo costo de operación.
- Conformación a estándares ANSI.
- Estrategia de almacenamiento MVCC para grandes volúmenes
- Soporta replicación de bases de datos.
- Interfases nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, XML.
- Soporta SSL nativo.

Plataformas que soporta:

- ✓ Corre bajo cualquier plataforma UNIX.
- ✓ Existe una versión para Windows.

MySQL

Características principales:

- Soporta los estándares ANSI.
- Contiene esquemas de almacenamiento independiente que se pueden seleccionar de acuerdo a las necesidades.
- Soporte para SSL.
- Querys con manejo de cache que puede incrementar el performance de la base de datos en un 200%.
- Permite manejo de replicación de bases de datos.
- Soporta indexado de texto.

Plataformas que soporta:

- ✓ Linux
- ✓ Windows.
- ✓ FreBSD
- ✓ Sun Solaris.
- ✓ IBM-AIX.
- ✓ Mac OS X.

Existen diversas clasificaciones de los RDBMS que se pueden realizar entre las cuales se encuentran:
Clasificación por ámbito:

- Comerciales (SQL Server, Sybase, Oracle, Informix, DB2)
- Software Libre (PostgreSQL, Mysql, Sybase (Linux)).

Por volumen de información:

- Corporativo (Oracle, Informix, Sybase, DB2, PostgreSQL)
- Departamental (SQL Server, SQL Anywhare, Mysql).

1.2.6 Tipos de datos usados por los RDBMS.

Los principales tipos de datos que se pueden encontrar en los manejadores de bases de datos son:

- Alfanuméricos.
- Numéricos.
- Booleanos.
- Fecha.
- Autoincrementables.

Aunque existe un estándar también en lo que se refiere a tipos de datos que deben de utilizar los RDBMS, estos sólo toman algunos e implementan otros. Esto se puede apreciar mejor en la tabla 1.4.

Tipo en SQL99	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server	Descripción
tinyint	tinyint			tinyint	tinyint	
smallint	smallint	smallint	smallint (lo convierte a number)	smallint	smallint	Entero con signo de 2 bytes
int, integer	int, integer	integer	int (lo convierte a number)	int	int	Entero con signo de 4 bytes
float()	float		float()	float	float	Número de punto flotante
double	double	double precision	double precision (lo convierte a float)	double precision	Double precision (se convierte en float)	Número Doble
real		real	real (Lo convierte a float)	real	real	Número Real
numeric(p,d)	numeric(p,d)	numeric(p,d)	number(p,d)	numeric(p,d)	decimal(p,d)	Numérico con precisión p y d decimales
character varying(n)	varchar(n)	varchar(n)	varchar2(n)	varchar(n)	varchar(n)	Carácter de longitud variable
char, character(n)	char(n)	char(n)	char(n)	char(n)	char(n)	Cadena de caracteres de longitud fija
date	date	date				Fecha sin hora del día
time	time	time				Hora del día
timestamp	timestamp	timestamp	date	datetime	datetime	Fecha y hora del día
boolean		boolean		bit	bit	Valor booleano
blob	blob	bytea	blob	image	image	Binary large object
clob	text	text	clob	text	text	Character large object
Interval		interval				Intervalo de tiempo

Tabla 1.4.- Tipos de datos que utilizan los RDBMS.

1.3.-SQL (STRUCTURED QUERY LANGUAGE).

SQL (Structured Query Language; Lenguaje Estructurado de Consulta) es un lenguaje de consulta para bases de datos que fue propuesto por Edgar F. Codd, siendo adoptado como estándar de la industria en 1986. Desde entonces se han realizado revisiones al estándar para incorporar nueva funcionalidad conforme la industria de las bases de datos lo va requiriendo. Una de las revisiones más importantes fue la de 1992, conocida como ANSI SQL92.

Actualmente la versión soportada por la mayoría de las bases de datos es el ANSI SQL99 también conocido como SQL3.

La ventaja de la adopción del ANSI SQL, es que los diversos RDBMS (Relational Data Base Management System; Sistema Manejador de Bases de Datos Relacional) tienen que acoplarse al estándar, permitiendo así una mayor compatibilidad entre ellos. Esto implica que conociendo una variante del SQL, se tienen los conocimientos necesarios para poder utilizar otros RDBMS: MS SQL Server, Oracle, Sybase, MySQL, PostgreSQL, DB2, etc.

Aunque los distintos fabricantes tratan de acoplarse al estándar ANSI SQL, es cierto que cada uno implementa funcionalidades extra que le dan un valor agregado a su producto pero sacrificando un poco la compatibilidad, por lo cual se podrán notar ciertas diferencias entre distintos RDBMS.

SQL es un lenguaje fácil de entender ya que su estructura utiliza palabras en inglés, lo que lo hace fácil de aprender y utilizar y las instrucciones se enfocan a qué buscar, dejando al RDBMS la tarea de cómo recuperar la información.

1.3.1 Definición de datos.

La tabla es el elemento fundamental de una base de datos relacional, la cual consiste de una serie de renglones (registros) que representan la información. Cada renglón está dividido en columnas (campos) los cuales deben de tener un tipo de dato establecido.

Tipos de datos del sistema y tipos de datos creados por el usuario.

Cada columna dentro de una tabla debe tener asociado un tipo de dato, siendo la labor del diseñador de la base de datos, el de encontrar el mejor tipo de dato que satisfaga las necesidades de almacenamiento y recuperación de cierta información.

Los tipos de datos que se manejan en una base, pueden variar ligeramente entre diferentes RDBMS, sin embargo el estándar ANSI, asegura que cierto tipo de datos estará presente en cualquier RDBMS asegurando así la compatibilidad.

Algunos RDBMS implementan sinónimos para los tipos de datos, de manera que puedan cumplir con el ANSI SQL99, sin embargo internamente son convertidos a un tipo de dato que si esté soportado.

Para ver los tipos de datos que soporta cada uno de los RDBMS ver la tabla 1.4.

Algunos RDBMS permiten crear tipos de datos a los usuarios para que cumplan con ciertas características con las que debe cumplir el sistema que se esta realizando.

Es importante conocer el concepto de valor nulo, en el contexto de una base de datos, debido a que frecuentemente un valor nulo es confundido con un valor numérico de 0 o una cadena vacía. Un valor nulo se representa en SQL con la cláusula NULL y representa la ausencia de información.

Tablas.

Como ya se menciono las tablas son el elemento fundamental que compone a una base de datos relacional porque todo gira en torno a ellas. Las tablas son estructuras de almacenamiento que albergan la información en forma de registros (renglones) que deben de ser identificados de manera única y esto se logra a través de una llave primaria.

La tabla es la representación física en la base de datos de una Entidad mientras que las relaciones son representadas mediante restricciones. En la siguiente figura 1.13 se puede ver mejor como se conforma una tabla.

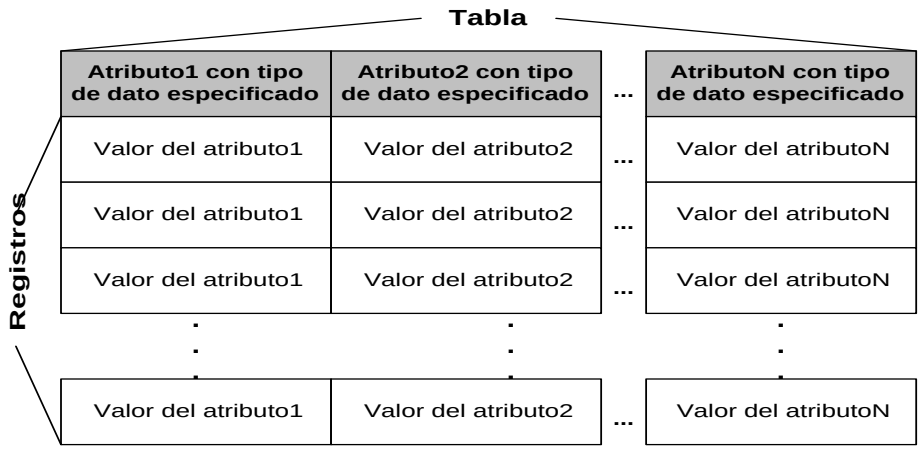


Fig. 1.13.- Representación gráfica de una tabla.

La sintaxis básica para la creación de una tabla es la siguiente:

```
CREATE TABLE <nombre_tabla>
(
<campo1> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
<campo2> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
<campo3> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
…
<campoN> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>]
)
```

La sintaxis general para agregar un campo es la siguiente:

```
ALTER TABLE <nombre_tabla>
ADD <campo> <tipo_dato> [DEFAULT <val_predeterminado>] [NOT NULL | NULL]
```

Cabe señalar que siempre que se agregue un campo a una tabla, éste debe permitir valores NULOS.

Para eliminar una tabla y los datos que contiene, así como sus índices, triggers y permisos se utiliza la siguiente instrucción:

```
DROP TABLE <nombre_tabla>
```

Reglas.

Las reglas dentro de la base de datos, permiten definir condiciones que debe cumplir la información para que sea válida. Por ejemplo se puede definir una regla que especifique que la percepción de un empleado no sobrepase los $10,000 pesos.

En Sybase y MS SQL Server las reglas se pueden crear como objetos independientes que se vinculan a distintas tablas. En cambio en Oracle y PostgreSQL no son objetos independientes y sólo pueden ser definidas como restricciones que afecta a una sola tabla.

Sintaxis para Sybase y Ms SQL Server:

```
CREATE RULE <nombre_regla>
AS <condicion>
```

Después de haber creado la regla, se le asigna al campo el cual queramos que cuente con esa regla.

```
sp_bindrule <nombre_regla>, 'nombre_tabla.nombre_columna'
```

Sintaxis para Oracle y PostgreSQL:

```
ALTER TABLE <nombre_tabla>
ADD CONSTRAINT <nombre_restriccion>
CHECK <condicion>
```

Defaults.

Los defaults establecen que valor será registrado de manera predeterminada para una columna, en caso de que no se especifique al momento de introducir los datos. En algunos RDBMS los defaults son objetos que se pueden emplear en diferentes tablas, mientras que en los demás, están ligados a la definición de la tabla. Al igual que con las reglas, la funcionalidad de los defaults pueden implementarse utilizando la lógica de la aplicación.

Nuevamente en Sybase y MS SQL Server es posible definir un DEFAULT como un objeto independiente que se puede vincular a varias campos de una o más tablas, mientras que en Oracle, MySQL y PostgreSQL los defaults están ligados a un solo campo.

Sintaxis en Sybase/Ms SQL Server para crear un default como objeto de la base de datos.

```
CREATE DEFAULT <nombre_default> AS <expresión_constante>
```

Una vez creado el default, con el procedimiento almacenado del sistema ligamos o unimos el default a una columna.

```
sp_binddefault <nombre_default> <nombre_tabla.nombre_columna>
```

Sintaxis para Oracle, PostgreSQL y Mysql:

```
CREATE TABLE <nombre_tabla>
(
 <campo1> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
 <campo2> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
)
```

Tipos de índices.

Un índice es una estructura de almacenamiento físico que permiten recuperar datos de una manera muy eficiente, su función principal es la de acelerar el proceso de recuperación de la información.

En un esquema relacional, cada registro dentro de una tabla debe de ser identificado de manera única, y esto se logra a través de una llave primaria, a la cual se le genera de manera automática un índice, que ayuda a ser más eficiente el proceso de consulta de la información. También existen las llaves foráneas, que son las columnas que hacen referencia a la llave primaria de otra tabla. A través de ellas se establecen relaciones entre tablas.

Tanto una llave primaria como una foránea establecen restricciones sobre el valor que pueda tener una columna. Las restricciones pueden tener un nombre asignado por el usuario o si este no se especifica, entonces el RDBMS será el encargado de asignarle un nombre interno a dicha restricción.

Llave primaria:

Una llave primaria permite identificar de manera única un registro dentro de una tabla. Al crear una llave de este tipo se genera automáticamente un índice de valores únicos, por lo que los valores de los campos que involucra la llave primaria no se pueden repetir ni ser nulos. Existen 3 formas de declarar una llave primaria, dos formas cuando se crea la tabla y una cuando ya se encuentra creada esta:

1.- Al crear una tabla se puede especificar que campo o campos forman parte de la llave primaria.

```
CREATE TABLE pais
(
  id_pais numeric(3) NOT NULL PRIMARY KEY,
  nombre varchar(100)
)
```

En el ejemplo anterior, la tabla solamente consta de un campo que forma la llave primaria “id_pais”, por ello es posible utilizar la cláusula PRIMARY KEY. Pero cuando la llave primaria consta de dos o más campos la sintaxis anterior no es útil, por lo cual se emplea la siguiente sintaxis:

```
CREATE TABLE pedido
(
  id_cliente numeric(10) NOT NULL,
  id_producto numeric(10) NOT NULL,
  fecha date,
  CONSTRAINT pedido_pk PRIMARY KEY (id_cliente, id_producto))
```

En la anterior sintaxis se puede observar que se ocupa la cláusula PRIMARY KEY, como si fuera un atributo más de la tabla, especificando entre paréntesis todos los campos que formen parte de la llave primaria.

2.- Si la tabla ya existe y se desea especificar los campos que forman parte de la llave primaria se emplea la cláusula ALTER TABLE.

```
ALTER TABLE pedido
ADD CONSTRAINT pedido_pk PRIMARY KEY (id_cliente, id_producto)
```

Llave foránea:

Las llaves foráneas son atributos de una tabla que hacen referencia a la llave primaria de otra tabla. Estas llaves foráneas permiten establecer relaciones entre las distintas tablas que existen dentro de la base de datos. La mayoría de los RDBMS implementan restricciones (constraints) cuando se genera una llave foránea, de este modo asegura la integridad de la información almacenada en la base de datos.

Las llaves foráneas se pueden crear dentro de la definición de la tabla, o una vez que esta ya existe, se puede utilizar la cláusula ALTER TABLE para agregar esta restricción. A diferencia de las llaves primarias, las llaves foráneas no generan un índice, por lo que de ser necesario se deberá crear con la cláusula CREATE INDEX.

1.- En el momento de crear la tabla:

```
CREATE TABLE resultado (
  id_pais1 number(3) NOT NULL,
  id_pais2 number(3) NOT NULL,
  resultado varchar(15) NOT NULL,
  CONSTRAINT pais_01_fk FOREIGN KEY (id_pais1) REFERENCES pais(id_pais),
  CONSTRAINT pais_02_fk FOREIGN KEY (id_pais2) REFERENCES pais(id_pais)
)
```

2.- A través de un ALTER TABLE:

```
ALTER TABLE resultado
  ADD CONSTRAINT pais_01_fk FOREIGN KEY (id_pais1) REFERENCES pais(id_pais)

ALTER TABLE resultado
  ADD CONSTRAINT pais_02_fk FOREIGN KEY (id_pais2) REFERENCES pais(id_pais)
```

Como ya se dijo, cuando se crea una llave foránea no se crea un índice implícito como con las llaves primarias, así que si se quisiera crear un índice a un campo la sintaxis sería la siguiente:

```
CREATE [UNIQUE] INDEX <nombre_índice>
ON <nombre_tabla>(<campo>,...)
```

Abusar del empleo de índices puede llevar a que se degrade el tiempo de respuesta del servidor en lugar de mejorarlo, esto se debe a que en operaciones que involucran inserción, modificación o eliminación de datos, los índices deben de ser actualizados lo cual puede consumir un tiempo considerable.

Si se quisiera eliminar algún índice se tendría que utilizar la siguiente instrucción:

```
DROP INDEX <nombre_índice>
```

1.3.2 Manipulación de datos.

La manipulación de datos sirve para poder administrar de una manera más eficiente los datos, ya sea obteniendo ciertos campos que se quieran analizar, actualizando los datos que han cambiado, borrar los que ya no se utilicen o insertar nuevos datos.

La mayor parte del trabajo con SQL girará entorno a cuatro comandos:

- *SELECT.* Permite seleccionar (recuperar) información de una tabla.
- *INSERT.* Permite agregar información a una tabla.
- *DELETE.* Permite eliminar información de una tabla.
- *UPDATE.* Permite actualizar información que existe en una tabla.

Para el empleo de cualquiera de los comandos mencionados es indispensable tomar en cuenta dos puntos:

- Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.
- Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

Selección de datos.

Las tablas dentro de una base de datos son las estructuras que tienen almacenada la información en forma de registros. Para poder recuperar esa información almacenada, se requiere del comando SELECT de SQL.

El comando SELECT es sumamente útil, ya que a través de él es posible realizar desde una consulta simple que sólo involucra una tabla, hasta una consulta compleja donde intervienen dos o más tablas, varias condiciones, agrupaciones de datos y ordenamientos.

Lo que se puede apreciar en la estructura de la instrucción SELECT, es que nunca debe faltar ni la palabra SELECT, ni FROM. Todos los demás elementos son opcionales.

Cláusula SELECT

Esta cláusula indica que la instrucción a ejecutar es una consulta a la base de datos. SELECT permite indicar el nombre de los campos que se quieren mostrar en la consulta. En caso de querer mostrar todos los campos de una tabla se emplea el comodín asterisco: *.

Cuando se realiza una consulta que involucra dos o más tablas, al nombre de cada campo se le antepone el de la tabla a la que pertenece.

```
<nombre_tabla>.<nombre_campo>
```

Es posible utilizar seudónimos (alias) para cambiar el nombre de las columnas mostradas en una consulta, esto puede servir para hacer más legible los resultados mostrados, o porque así lo requiere alguna aplicación. Para colocar los seudónimos es necesario especificarlo mediante la cláusula AS, de la siguiente forma:

```
<nombre_campo> AS <otro_nombre>
```

Cláusula FROM.

La cláusula FROM sirve para indicar las tablas de las cuales se desea mostrar la información. Cuando una consulta involucra dos o más tablas, es indispensable establecer las relaciones que existen entre ellas (join) mediante una cláusula WHERE.

Cláusula WHERE.

La cláusula WHERE permite delimitar los registros que serán mostrados en la consulta, a través de criterios o condiciones. Es posible utilizar los operadores lógicos: OR, AND y NOT para combinar expresiones y refinar el criterio de consulta.

Para escribir condiciones de manera adecuada es muy importante recordar que:

- Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.
- Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

El valor NULL es un valor especial por lo cual se debe tener sumo cuidado cuando se desee utilizar condiciones con NULL. La única forma de comparar contra un valor NULL es utilizar el operador IS o IS NOT.

```
SELECT * FROM empleado WHERE comision = NULL;    --incorrecto
SELECT * FROM empleado WHERE comision is NULL;   --correcto
```

Las expresiones más frecuentes son las que involucran una comparación entre dos elementos como se muestra en la tabla 1.5.

Igualdad	=	empleado.id_departamento = departamento.id_departamento
Desigualdad	<>ó!=	nombre_cargo != 'Director'
Mayor que	>	sueldo > 15000
Menor que	<	edad < 35
Mayor o igual que	>=	sueldo >=15000
Menor o igual que	<=	edad <= 35
Similar a	LIKE	nombre_empleado like 'Al%' (% es un comodín)
Es	IS	edad IS NULL
No es	IS NOT	edad IS NOT NULL

Tabla 1.5.- Operadores de comparación.

En SQL es posible abreviar la forma de escribir ciertas condiciones como se puede apreciar en la tabla 1.6.

La comparación de similitud que se hace mediante el uso de la cláusula LIKE, requiere de incluir comodines, que sustituyan uno o varios caracteres.

%	representa 0 o más caracteres.
_	representa 1 carácter

La expresión:	Se simplifica usando:	Quedando de la siguiente manera:
sueldo >= 10000 AND sueldo <=15000	**BETWEEN**	`Sueldo BETWEEN 10000 AND 15000`
nombre_cargo = 'Gerente' OR nombre_cargo = 'Presidente' OR nombre_cargo = 'Vicepresidente' OR nombre_cargo = 'Director'	**IN**	`nombre_cargo IN ('Gerente', 'Presidente', 'Vicepresidente', 'Director')`
nombre_cargo != 'Jefe de departamento' AND nombre_cargo != 'Vendedor'	**NOT IN**	`nombre_cargo NOT IN ('Jefe de departamento', 'Vendedor')`

Tabla 1.6.- Otros operadores de comparación.

Cláusula GROUP BY.

En la cláusula GROUP BY se indica el o los campos por los cuales se desea agrupar un conjunto de registros. Comúnmente esta agrupación va acompañada con una serie de funciones que realizan ciertas operaciones sobre el valor de los campos indicados. Estas funciones son conocidas como funciones de agregación o agrupación las cuales se muestran en la tabla 1.7.

Función	Acción
`COUNT(*)`	Regresa el número de registros encontrados
`COUNT(<campo>)`	Regresa el número de registros cuyo valor del campo especificado no es nulo
`SUM(<campo>)`	Suma los valores de la columna especificada
`AVG(<campo>)`	Promedia los valores del campo especificado
`MIN(<campo>)`	Regresa el valor mínimo del campo especificado
`MAX(<campo>)`	Regresa el valor máximo del campo especificado

Tabla 1.7.- Funciones de agregación.

Cláusula HAVING.

Esta cláusula es el equivalente a la cláusula WHERE, es decir, especifica un criterio o condición, pero la diferencia radica en que se ocupa únicamente cuando se desea especificar una función de agregación en la condición.

Cláusula ORDER BY

Esta cláusula permite indicar los campos por los cuales se desea ordenar la información mostrada. Es posible indicar si el orden es descendente o ascendente, de manera predeterminada es ascendente. Algunos RDBMS permiten realizar este ordenamiento especificando, en lugar del nombre del campo, la posición de este.

Inserción de datos.

A través de la instrucción INSERT de SQL, se introduce la información a una tabla.

La estructura general de este comando es la siguiente:

```
INSERT INTO <tabla> [(<nombreCampo1>, <nombreCampo2>, <nombreCampo3> ...)]
{VALUES (<valorCampo1>, <valorCampo2>, <valorCampo3> ... ) | <Expresión select> }
```

Cláusula INSERT.

Esta cláusula permite indicar que la operación a realizar es la inserción de un registro.

Cláusula INTO.

Esta cláusula permite indicar la tabla en donde se realizará dicha inserción. Únicamente se puede especificar una tabla a la vez. Después del nombre de la tabla puede o no ir el nombre de los campos donde se va insertar información, esto es opcional, pero es muy recomendable no omitirlos, porque le resta legibilidad a la instrucción.

Si no se especifica el nombre de los campos que se van a insertar, el DBMS identifica que se desea insertar información en cada uno de los campos, en el orden definido por la estructura de la tabla.

Cláusula VALUES.

Esta cláusula permite especificar los valores a insertar para cada uno de los campos involucrados en la sentencia.

Para especificar los valores a insertar de manera adecuada es muy importante recordar que:

- Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.
- Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

Eliminación de registros.

La instrucción DELETE elimina registros de la tabla indicada con la posibilidad de indicar un criterio, en caso de omitirlo, se eliminan todos los registros de la tabla.

La sintaxis es la siguiente:

```
DELETE FROM <nombre_tabla>
[ WHERE <condición> ]
```

Cláusula DELETE.

La cláusula DELETE permite indicar que la operación a realizar es una eliminación de registros.

Cláusula FROM

La cláusula FROM permite especificar la tabla de donde se desea eliminar registros.

Cláusula WHERE

La cláusula WHERE permite delimitar el conjunto de registros a eliminar, si no se especifica esta condición, se eliminaran todos los registros que contenga la tabla especificada.

Actualización de datos.

Para modificar o actualizar los valores de los registros de una tabla se utiliza el comando UPDATE. Si no se especifica una condición con la cláusula WHERE, todos los registros que existan en la tabla serán actualizados.

La sintaxis es la siguiente:

```
UPDATE <nombre_tabla>
SET <campo1> = <valor1>, <campo2> = <valor2>, ....
[ WHERE <condición> ]
```

Cláusula UPDATE.

La cláusula UPDATE es la que indica que la operación a ejecutar es una actualización. Después de la cláusula se especifica el nombre de la tabla en donde se encuentra la información que deseamos modificar. Sólo se puede especificar una tabla a la vez.

Cláusula SET.

Esta cláusula permite especificar los campos que se desean modificar y su nuevo valor. La cláusula se coloca una sola vez aunque sean varios campos los que se deseen modificar.

Cláusula WHERE.

Esta cláusula permite especificar un criterio para delimitar el conjunto de registros a modificar.

Vistas.

Una vista es una tabla qué no ocupa espacio de almacenamiento para la información que contiene, porque su estructura e información está definida a través de la ejecución de una instrucción SELECT. Las vistas tiene dos usos: el primero es para simplificar el acceso a datos que se ocupan frecuentemente y que requieren una sentencia de SQL muy compleja para dicho acceso; y el segundo es con fines de seguridad, que permitan mantener ocultas ciertas columnas.

La sintaxis para crear una vista es:

```
CREATE VIEW <nombre_vista> AS <instrucción SELECT>
```

Un ejemplo básico para conocer su funcionamiento seria:

```
CREATE VIEW empleados_h
AS SELECT * FROM empleados
WHERE sexo = 'H'
```

Para eliminar una vista se utiliza la siguiente instrucción:

```
DROP VIEW <nombre_vista>
```

Definición de privilegios.

Unos de los componentes de un RDBMS es el DCL (Data Control Language) que permite controlar y establecer restricciones de acceso a la información contenida en la base de datos. Es tarea del administrador de la base de datos el encargado de asignar o revocar permisos y/o crear usuarios en la base de datos. El manejo de usuario varía considerablemente de un RDBMS a otro, siendo que en algunos es más completo, el esquema del manejo de usuario y permisos, que en otros.

Donde como ya se dijo GRANT se utiliza para otorgar privilegios y REVOKE para eliminarlos.

1.3.3 Funciones de utilidad.

Las funciones de utilidad son funciones que ya están predefinidas y con las cuales debe contar los RDBMS, aunque son un estándar del SQL los manejadores tienen muchas diferencias entre estas, o las funciones son llamadas de diferente forma que el estándar.

Funciones para datos tipo caracter.

En la tabla 1.8 se muestran las funciones de tipo caracter que se especifican en el estándar.

Función	Descripción	Ejemplo
char_length(<*cadena*>) / character_length(<*cadena*>)	Determina la longitud de la cadena especificada	char_length('jose')
lower(<*cadena*>)	convierte el texto a minúsculas	lower('TOM')
octet_length(<*texto*>)	almacena el tamaño del texto	octet_length('jose')
position(<*cadena1*> in <*cadena2*>)	posición de un subtexto especificado	position('o' in 'Tom')
substring(<*cadena*> [from <*entero*>] [for <*entero*>])	extrae un subtexto especificado	substring('Tom' from 2 for 2)
trim([leading|trailing|both] [<*cadena1*>] from <*cadena2*>)	remueve caracteres de un texto	trim(both 'x' from 'xTomx')
upper(<*cadena*>)	convierte un texto a mayúsculas	upper('tom')
<*cadena1*> || <*cadena2*>	Concatenación de cadenas	'Hola' || 'Mundo'
Case WHEN <*expresion*> THEN <*valor1*> [ELSE <*valor2*>] END	Evalúa la expresión si es verdadera regresa valor1 de lo contrario regresa valor2	CASE WHEN id_departamento is null THEN 'No tiene asignado Departamento' END

Tabla 1.8.- Funciones para datos de tipo carácter.

Funciones matemáticas.

Estas funciones implementan funciones matemáticas para poder utilizarlos con los datos que se tienen y en la tabla 1.9 se muestran las funciones estándar y algún nombre de función equivalente en los diferentes manejadores.

SQL92	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server
abs	Si	Si	Si	Si	Si
acos	Si	Si	Si	Si	Si
asin	Si	Si	Si	Si	Si
atan	Si	Si	Si	Si	Si
ceiling	Si	ceil	ceil	Si	Si
cos	Si	Si	Si	Si	Si
cot	Si	Si	No	Si	Si
degrees	Si	Si	No	Si	Si
floor	Si	Si	Si	Si	Si
log	Si	Ln	Ln	Si	Si
log10	Si	log	Log	Si	Si
pi	Si	Si	No	Si	Si
power	Si	pow	Si	Si	Si
radians	Si	Si	No	Si	Si
rand	Si	random	No	Si	Si
round	Si	Si	Si	Si	Si
sign	Si	Si	Si	Si	Si
sin	Si	Si	Si	Si	Si
sqrt	Si	Si	Si	Si	Si
tan	Si	Si	Si	Si	Si

Tabla 1.9.- Funciones matemáticas.

Funciones para datos tipo fecha.

Las funciones estándar para manejo de fecha son las que se muestran en la tabla 1.10.

Función	Descripción
Current_date / Current_date()	Regresa la fecha actual en formato 'YYYY-MM-DD' o YYYYMMDD
Current_time / Current_time()	Regresa la hora actual en formato 'HH:MM:SS' o HHMMSS
Current_timestamp / Current_timestamp()	Regresa la fecha y hora actual con formato 'YYYY-MM-DD HH:MM:SS' o YYYYMMDDHHMMSS

Tabla 1.10.- Funciones para datos tipo fecha.

Desafortunadamente ni Oracle, ni Sybase, ni MS SQL Server implementan las funciones definidas por el estándar, manejando sus propias funciones, y los campos tipo fecha incluyen también la hora como si fueran un campo de tipo timestamp.

1.3.4 Manejo de transacciones.

Los RDBMS deben de implementar un mecanismo a través del cual, los cambios a realizar en la información, a través de una instrucción INSERT, DELETE o UPDATE, no son efectuados hasta que el usuario lo indique explícitamente. Una transacción puede comprender una o más instrucciones SQL.

Una transacción es atómica, porque todas las instrucciones de SQL deben completarse con éxito, o ninguna de ellas. Una vez que el RDBMS determina que la transacción fue exitosa es necesario que la información sea almacenada de manera permanente. Una base de datos transaccional garantiza que todas las operaciones realizadas en una transacción sean guardadas en almacenamiento permanente antes de que ésta sea reportada como completada, previniendo así, perdida de información por fallas del equipo, por ejemplo en un corte del suministro de energía.

Cuando múltiples usuarios realizan transacciones de manera concurrente, cada uno de ellos no debe ver los cambios incompletos realizados por los demás. En el momento que la transacción finaliza adecuadamente y es almacenada permanentemente, los cambios se vuelven visibles para todos los demás usuarios.

Commit y rollback.

El comando *COMMIT* permite indicar que los cambios a realizar dentro de una transacción sean llevados a cabo de manera permanente, y el comando *ROLLBACK* permite deshacer los cambios. El RDBMS reserva un espacio de almacenamiento, donde registra todas las instrucciones de SQL que se deben de ejecutar. De esta manera es posible deshacer los cambios realizados por operaciones UPDATE, DELETE o INSERT.

Las transacciones se inician de distinta manera, aunque similar. Por ejemplo en PostgreSQL, una transacción es iniciada mediante la cláusula BEGIN y en Sybase se emplea BEGIN TRANSACTION, seguida de las instrucciones de SQL a realizar y finalmente se emplea la cláusula COMMIT para indicar que las modificaciones deben realizarse de manera permanente o en caso que se desee cancelar la operación, se ocupa la cláusula ROLLBACK.

Commit y rollback por fases.

Es posible especificar un punto en la transacción al cual se puede posteriormente realizar un rollback sin que esto afecte a toda la operación sino únicamente hasta el punto de referencia indicado (SAVEPOINT).

Este tipo de transacciones con rollback por fases, no está disponible en todos los RDBMS.

1.3.5 Estructuras de control de flujo.

Este tipo de estructuras se emplea en el desarrollo de programas, que en un RDBMS puede ser un procedimiento almacenado[17]. Los RDBMS comerciales son los que se han desarrollado más en el aspecto de programación en la base de datos, sin embargo las instrucciones aunque similares, difieren entre uno y otro.

If-then-else-end if

Ésta estructura de control de flujo se utiliza como bifurcación, es decir, se tienen dos posibilidades y se decide cual de las dos posibilidades se va a ejecutar.

[17] Los procedimientos almacenados son funciones que realizan operaciones, estos pueden ser del sistema o hechos por el usuario.

For loop.

La sintaxis en Oracle es la siguiente:

```
FOR <variable> IN <valor inicial>...<valor final> LOOP
   <lista_de_instrucciones>
END LOOP;
```

Este ciclo iterativo inicializa la variable especificada y ejecuta las instrucciones cíclicamente hasta que la variable alcanza el valor final.

While loop.

Esta estructura se utiliza cuando se desea realizar una o un grupo de instrucciones repetidamente. La sintaxis en Sybase y MS SQL Server es:

```
WHILE <expresión>
<instrucciones a ejecutar mientras la expresión sea verdadera>
      [BREAK]
      [CONTINUE]
```

Donde BREAK se utiliza para salir del ciclo en el punto donde esta palabra se encuentra y CONTINUE se utiliza para regresar directamente a evaluar nuevamente la expresión.

En Oracle la sintaxis es la siguiente:

```
WHILE <condicion> LOOP
   <lista_de_instrucciones>
END LOOP;
```

1.3.6 Procedimientos almacenados.

Un procedimiento almacenado es un conjunto de comandos de SQL que pueden ser compilados y almacenados en el servidor. Una vez realizado esto, los clientes no necesitan volver a teclear todas las instrucciones sino únicamente hacer referencia al procedimiento. Esto mejora el rendimiento del servidor, ya que la instrucción de SQL solamente es revisada una sola vez y menos información debe ser enviada entre el cliente y el servidor.

Declaración de variables.

Las variables son elementos fundamentales en la programación de procedimientos. Las variables deben ser declaradas al inicio del programa, antes de utilizarlas, para ello existe la cláusula DECLARE en Sybase, Ms SQL Server y Oracle.

Las variables locales sólo existen durante la ejecución del procedimiento donde son declaradas; cuando éste termina, las variables locales son eliminadas.

En Sybase y Ms SQL Server se debe observar lo siguiente para el manejo de variables:

- Los nombres de las variables locales deben estar precedidos invariablemente por una @

- Los nombres de las variables globales deben estar precedidos invariablemente por @@
- Antes de utilizar una variable ésta debe ser declarada indicando el nombre y el tipo de dato de la misma.

Para declarar las variables se utiliza la cláusula DECLARE, de la siguiente manera:

```
DECLARE  <@variable> <tipo de dato> <(tamaño)> [, .....]
```

Un ejemplo de esto es:

```
DECLARE @nombre char(50),  @edad smallint, @sueldo numeric(10,2)
```

- Las variables deben ser declaradas al inicio del batch
- Para asignar valores a una variable se utiliza la instrucción SELECT, de la siguiente forma:

```
SELECT <@variable> = <expresión> [,...]
[FROM ....]
[WHERE...]
```

El uso de las cláusulas FROM y WHERE es opcional.

Si una instrucción SELECT que recupera datos de una tabla no regresa registros, las variables ahí involucradas conservan su valor.

Creación de Procedimientos.

Para crear procedimientos almacenados se utiliza la siguiente instrucción en Sybase y MS SQL Server:

```
CREATE PROCEDURE <nombre_procedimiento>
[(@<nombre_parámetro> <tipo_de_dato>)]
AS
      <instrucciones SQL>
RETURN
```

Ejecución de Procedimientos.

Para ejecutar un procedimiento almacenado en Sybase ó SQL Server, sólo se tiene que indicar su nombre, en caso de que sea la primera instrucción dentro de un programa. En caso contrario se antepone la palabra exec. Si el procedimiento almacenado recibiera parámetros, éstos deben de indicarse después del nombre.

```
[EXEC]  <nombre_procedimiento> [<parámetro>, ...]
```

Eliminación de Procedimientos.

Para eliminar un procedimiento almacenado, se dispone de la instrucción DROP PROCEDURE. La sintaxis es la siguiente:

```
DROP PROCEDURE <nombre_procedimiento>
```

Paso de Parámetros.

El uso de parámetros incrementa la flexibilidad de un procedimiento almacenado. Éstos se definen desde la creación del procedimiento y deben ser proporcionados por el usuario al momento de ejecutarlo.

Un ejemplo en Sybase de un procedimiento con parámetros es el siguiente:

```
CREATE PROCEDURE sueldo_empleado
@sueldo numeric(8,2)
AS
     SELECT nombre, sueldo
     FROM    empleado
     WHERE  sueldo <= @sueldo
     ORDER BY nombre
RETURN
```

Para ejecutar el procedimiento:

```
EXEC sueldo_empleado 10000
```

1.3.7 Triggers

Un trigger es un procedimiento almacenado que es invocado cuando un evento en particular ocurre. Por ejemplo, se puede ejecutar (disparar) un procedimiento almacenado cada vez que se borre, actualice o inserte un registro. Los procedimientos almacenados que se invocan tiene la restricción de que no pueden manejar parámetros ni ser invocados directamente.

Los TRIGGERS no forman parte del estándar SQL92, pero fueron contemplados posteriormente en el SQL99. La mayoría de los RDBMS tienen soportes para triggers, pero algunos otros no.

En Sybase y SQL Server un trigger al dispararse crea dos tablas temporales como se ve en la tabla 1.11 las cuales sólo pueden ser accesadas dentro del trigger, y poseen la misma estructura de la tabla a la que está ligada. Estas tablas son: Inserted y Deleted.

Tabla	Contenido	En Triggers
Inserted	Contiene los registros que se van a agregar a la tabla, como resultado de los comandos Insert y Update	Insert, Update
Deleted	Contiene los registros que se van a eliminar de la tabla, como resultado de los comandos Delete y Update	Delete, Update

Tabla 1.11.- Tablas temporales creadas por el trigger.

En Oracle se utiliza el alias OLD y NEW para hacer referencia a los valores antes de la actualización y el nuevo valor, respectivamente.

Creación de triggers.

Para crear un trigger se utiliza la siguiente sintaxis como se observa en la tabla 1.12.

Sybase/Ms SQL Server	Oracle
CREATE TRIGGER <*nombre_trigger*> **ON** <*nombre_tabla*> **FOR** [**INSERT** | **UPDATE** | **DELETE**] **AS** <*instrucciones SQL*> ... **RETURN**	**CREATE** [**OR REPLACE**] **TRIGGER** <*nombre_trigger*> {**BEFORE**|**AFTER**} {**INSERT**|**DELETE**|**UPDATE**} [**OF COLUMN** <*nombre_columna*>] **ON** <*nombre_tabla*> [**FOR EACH ROW**] <*codigo pl/sql*>

Tabla. 1.12.- Creacion de un trigger en Sybase y Oracle.

Un ejemplo que se puede utilizar para comprender mejor la creación de un trigger puede ser el que a continuación se representa, este ejemplo será realizado con la sintaxis de Sybase/SQL Server.

Eliminación de triggers.

La sintaxis para eliminar un trigger es muy simple, únicamente se utiliza la instrucción DROP TRIGGER.

DROP TRIGGER <*nombre_trigger*>

1.4.-ACCESO A DATOS A TRAVÉS DE LA PROGRAMACIÓN DE CLIENTES.

El acceso a los datos a través de clientes es una manera de manejar nuestros datos de una buena forma, además de poder acceder a ellos en una forma más fácil.

Los clientes son programas que facilitan el acceso a los datos de una forma transparente para el usuario, estos clientes son programados con lenguajes de programación diversos, pueden ser clientes que sólo se puedan ejecutar en una máquina donde se encuentren instalados u otra posibilidad puede ser que el cliente este el Web[18].

1.4.1 Uso de PHP.

PHP es un lenguaje de programación que permite la generación dinámica de contenidos en un servidor Web. Su nombre oficial es PHP: “Hypertext Preprocessor”.

PHP permite embeber pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas íntegramente en un lenguaje distinto al HTML.

Características:

- Es un potente y robusto lenguaje de programación embebido en documentos HTML.
- Dispone de librerías de conexión con la gran mayoría de los manejadores de base de datos.

[18] La Web es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet.

- Proporciona soporte a múltiples protocolos de comunicaciones en Internet (HTTP, FTP, etc.).

Algunas ventajas del lenguaje son:

- Código fuente abierto: Permite posibles mejoras o sugerencias acerca de su desarrollo (PHP ha sido escrito en lenguaje C).
- Gratuito: No es necesario realizar ningún desembolso económico.
- Portable y multiplataforma: Versiones para múltiples plataformas (Windows 95, 98, NT, 2000, Unix, Linux, etc.).
- Eficiente: Consume muy pocos recursos en el servidor
- Alta velocidad de desarrollo: Proporciona gran cantidad de librerías muy útiles y bien documentadas que ahorran mucho trabajo al programador.
- Capacidad de conexión con la mayoría de los manejadores de base de datos.
- No requiere definición de tipos de variables.

Funcionamiento:

- Es un lenguaje que se ejecuta en el servidor: no es necesario que su navegador lo soporte.
- El servidor donde están alojadas debe soportar PHP, es decir, tener instalado el intérprete de PHP.
- Se recomienda tener también un manejador de base de datos para aprovechar al máximo las funcionalidades que PHP ofrece.

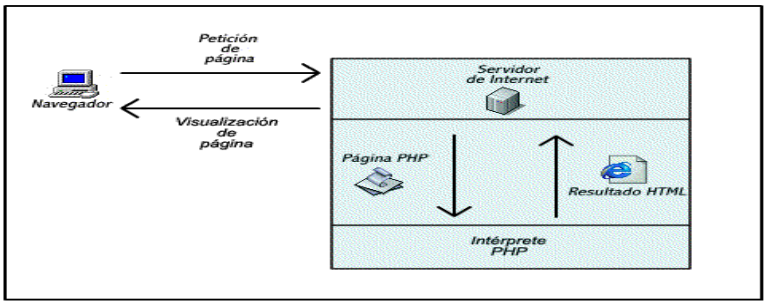


Fig. 1.14.- Funcionamiento de PHP.

Sintaxis básica:

- PHP es un lenguaje embebido en HTML.
- Es necesario indicarle al intérprete cuales son las partes escritas en PHP
- Esto se hace delimitando nuestro código por etiquetas.
- Las etiquetas soportadas por PHP son: <?php ?>

1.4.2 Uso de JSP.

La tecnología de Java Server Pages (JSP)[19] permite a los desarrolladores y a los diseñadores de Web desarrollar rápidamente y mantener fácilmente páginas dinámicas, ricas en información como son las que soportan a sistemas de negociación. La tecnología de los JSP separa la interfaz del usuario de la parte lógica del contenido permitiendo a los diseñadores cambiar a su disposición las plantillas de la interfaz sin alterar el contenido de la programación.

[19] JSP es parte de la tecnología de Java la cual se utiliza para construir sistemas Web´s.

Por lo tanto, se puede decir que es una tecnología de Java que permite introducir código para la generación dinámica de HTML dentro de una página Web. Esta surge por la necesidad de crear aplicaciones dinámicas para Web de forma fácil, ya que la mayor parte del resultado de un programa CGI es estático. Se podría pensar entonces en JavaScript[20], pero éste genera HTML dinámicamente en el cliente y no puede acceder a los recursos del servidor. La ventaja como ya se dijo es que permite diseñar la página Web y el código de forma independiente. Las páginas JSP pueden residir en cualquier parte del servidor.

En otras palabras, un JSP es simplemente una página de Web en HTML que contiene dígitos binarios adicionales de código que ejecuta la lógica de la aplicación para generar contenido dinámico. Por ejemplo, un JSP puede contener código HTML que despliega texto y gráficos estáticos, y además puede invocar a métodos de un objeto de JDBC que tenga acceso a una base de datos; cuando se despliegue la página en el browser de un usuario, ésta contendrá el contenido estático HTML y la información dinámica extraídos de la BD.

La figura 1.15 da una visión más clara de cómo funciona JSP.

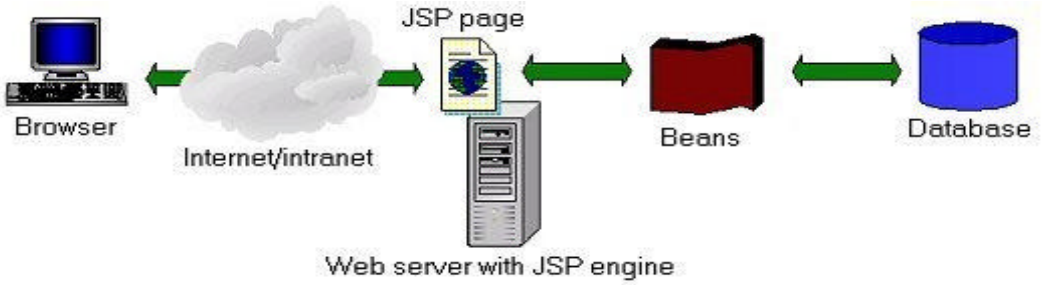


Fig. 1.15.- Funcionamiento de JSP.

Para realizar una petición de una página JSP se sigue una forma similar al de una página HTML estática, aunque se realizan otras actividades adicionales, las cuales son transparentes para el usuario.

Para una página no dinámica se teclea un URL en el browser y éste usando un protocolo HTTP[21] mandará una petición del archivo con extensión html (ej. archivo.html) a un servidor Web, posteriormente el servidor extraerá el archivo y lo mandará a el browser, el cual hace uso de las etiquetas de HTML del archivo para ser presentado al usuario final.

No así para una página dinámica, ya que en este caso se hace la petición de un archivo con extensión jsp (ej. archivo.jsp) y de este punto en adelante las cosas cambian empezando con que no sólo se puede hacer la petición desde un browser sino también desde otra página JSP, la petición llegará también a un servidor Web, el cual será capaz de reconocer la extensión jsp y aquí se compilara el código JSP, el cual una vez interpretado regresara una respuesta en formato html para que el browser lo pueda interpretar.

Características de JSP:

- Conjunta el poder de Java en el servidor y la flexibilidad de HTML (HyperText Markup Language) en el browser.
- No sólo se puede utilizar HTML, sino también XML (eXtensible Markup Language).

[20] Javascript es un lenguaje que se interpreta en el cliente, sirve para darle un mayor dinamismo a las páginas Web.
[21] HTTP es el protocolo de transferencia de hipertexto (*HyperText Transfer Protocol*), es usado en cada transacción de la Web, y el cual fue desarrollado por la W3C.

- Hace más fácil reusar componentes como JavaBeans[22] los cuales realizan tareas más específicas.
- Su función es saber como procesar una solicitud para crear una respuesta.
- Soporta contenido dinámico que refleja las condiciones del mundo real.
- Existe independencia entre la parte del diseño (interfaz) y la lógica (programa).
- Cuenta con independencia de plataforma.
- Ante un cambio se compila automáticamente.
- Es más rápido y fácil crear aplicaciones de Web.
- Capaz de instanciar cualquier clase de Java.
- Corre en todos los servidores de Web principales.

1.5.-FUNDAMENTOS DE SISTEMAS OPERATIVOS.

Una computadora sin software es un montón de fierros que no es de gran ayuda, necesita de programas para solucionar nuestros problemas. Estos programas pueden clasificarse en dos grandes grupos:

1. *Programas de sistema (software de sistema):* Controlan la operación de la computadora. Los más representativos son los compiladores y el sistema operativo.

2. *Programas de aplicación (software de aplicación):* Resuelven problemas para los usuarios. Algunos ejemplos pueden ser: procesadores de palabras, hojas de cálculo, manejadores de bases de datos, juegos, etc.

Un Sistema Operativo es un conjunto de programas que se encargan de la gestión de los recursos de la computadora como la memoria, las entradas y salidas, interrupciones, tiempos del procesador[23], de la seguridad, etcétera; además de servir de interfaz entre el usuario y el hardware.

Hardware

Es la parte física de la computadora, es la parte tangible como el disco duro, el cpu, la memoria, dispositivos de entrada y salida, así como otros medios de almacenamiento.

Software

Es la parte lógica de la computadora, lo que no se puede tocar como los datos, la información, los programas instalados.

Responsabilidades del administrador del sistema operativo

El administrador del sistema es responsable de que las operaciones básicas del sistema que se encuentren funcionando, entre las tareas de administración se encuentran las siguientes:

- Instalar el sistema operativo
- Administración de las cuentas de usuarios y grupos

[22] Un JavaBean o bean es un componente hecho en software que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.
[23] Un procesador es un circuito electrónico integrado que actúa como unidad central de proceso de una computadora, proporcionando el control de las operaciones y de los tiempos para los programas.

- Mantenimiento de la seguridad del sistema
- Configurar dispositivos
- Instalar y particionar discos
- Administrar paquetes de software
- Realizar tareas de respaldo y recuperación
- Configurar archivos de inicialización del sistema

Sistema Operativo Linux

Los tres principales componentes de un sistema operativo tipo Linux son:

- El kernel: Es el programa maestro del sistema operativo que administra los recursos del sistema.
- El shell: Es el intérprete de comandos, además de ser la interfaz entre el sistema operativo y el usuario.
- Árbol de directorios: Es el conjunto de carpetas donde se encuentra toda la información del sistema operativo.

Instalación de Linux (CentOS[24])

CentOS (acrónimo de Community ENTerprise Operating System) es un clon a nivel binario de la distribución Red Hat Enterprise Linux, compilado por voluntarios a partir del código fuente liberado por Red Hat.

Los principales que se utilizan para la instalación del sistema operativo centOS son:

1. Insertar el primer CD en el equipo y reiniciarla.
2. Después de reconocer el hardware del equipo, se despliega la pantalla inicial de instalación con el promt boot en la parte inferior de la pantalla.
3. Presionar ENTER o dejar pasar unos segundos para que inicie la instalación.
4. Como parte del proceso de inicio, CentOS genera un disco RAM que es usado en lugar de un disco duro. Las herramientas usadas para la instalación, son copiadas al disco RAM. El uso de este disco RAM permite establecer una configuración sin borrar o dañar la información contenida en el disco duro. Una vez validada la configuración se inicia la copia de archivos al disco duro.
5. Si se desea, se puede revisar la integridad de los CD de instalación.
6. Anaconda inicia y prueba el hardware para iniciar el modo gráfico.
7. Anaconda[25] colecta información acerca de cómo se desea realizar la instalación.
8. Cuando el proceso de Anaconda termina de recolectar información, advierte sobre el riesgo de perder información del disco duro.
9. Cuando el sistema reinicia, el script de instalación realiza una serie de preguntas antes de completar la instalación.
10. CentOS está listo para usarse.

Administración del sistema operativo.

Una importante tarea del administrador del sistema es configurar las cuentas de usuario proporcionando los recursos y servicios necesarios.

Las cuentas de usuario se conforman de un nombre de usuario y una contraseña principalmente.

[24] CentOS es una distribución de Linux basada en otra distribución llamada Red Hat.
[25] Anaconda es un instalador en ambiente gráfico el cual nos permite realizar de una forma más fácil la instalación del sistema operativo Linux en su versión de CentOS.

En un sistema Linux la información de las cuentas de usuarios se almacenan en los siguientes archivos: /etc/passwd, /etc/shadow.

Otra forma de administrar a los usuarios es por grupos, en donde cada grupo representa diferentes niveles y tipos de usuarios.

Usando Linux, todas las comunicaciones de red acontece entre interfaces, que son dispositivos de red conectados al sistema, configurados de modo determinado y usando un protocolo. Los archivos de configuración para las diferentes interfaces de red y scripts para activarlos están ubicados en el directorio /etc/sysconfig/network-scripts.

Los principales archivos de configuración de red son:

/etc/host : El principal propósito del archivo es resolver los nombres de hosts.

/etc/resolv.conf: Este archivo especifica las direcciones IP[26] de los servidores DNS[27] y el dominio de búsqueda.

/etc/sysconfig/network: Especifica la información del routing y del host para todas las interfaces de red.

/etc/sysconfig/network-scripts/ifcfg-<interface-name>: Para cada interfaz de red del sistema Linux existe un script de configuración de interfaz para una interfaz de red determinada.

Todo el software en un sistema Linux está dividido en paquetes RPM[28] los cuales pueden ser instalados, actualizados o eliminados.

RPM mantiene una base de datos de los paquetes instalados y de sus archivos. Tiene cinco modos de operación básicos: instalación, desinstalación, actualización, consulta y verificación.

Existe otro método para la instalación del software, el cual es a través de código fuente[29]. Éste es el método universal para todas las distribuciones de Linux.

Es necesario leer la documentación que acompaña a dicho paquete y seguir las instrucciones proporcionadas por el autor. Por lo general son necesarios al menos tres pasos:

- . /configure --prefix=/usr/local/
- make
- make install

Se puede crear y recuperar respaldos utilizando las herramientas gzip, zip, gunzip y tar.

[26] IP es un protocolo de Internet el cual sirve para reconocer en Internet a una computadora.
[27] DNS o servidor de nombres de dominio, es el encargado de cambiar las direcciones IP en nombres de páginas Web.
[28] RPM es el administrador de paquetes de Red Hat, es una herramienta para la administración de los paquetes de las distribuciones basadas en Linux.
[29] El código fuente es el programa en su forma original, tal y como fue escrito por el programador, el código fuente no es ejecutable directamente por la computadora, debe convertirse en lenguaje de maquina mediante compiladores, ensambladores o intérpretes.

Diagnóstico del sistema.

Se puede hacer uso de una gran gama de herramientas que vienen incluidas en la distribución de Linux para poder diagnosticar cuales son los problema que se presenta en el servidor.

Monitoreo de procesos

Se puede hacer uso de los siguientes comandos para ver las tareas que se están ejecutando en el sistema y la cantidad de recursos que ocupa cada uno de ellos. "*ps"* sirve para listar los procesos que están corriendo en el servidor.

Espacio en disco.

Los principales comandos para el diagnóstico de espacio en disco son:

df: Sirve para determinar cuanto espacio se esta utilizando en todas las particiones.
du: Se utiliza para determinar cuanto espacio tiene un archivo o directorio.

Monitoreo de la red.

Para monitorear la red se utilizan básicamente los siguientes comandos:

- ifconfig: permite configurar las interfaces de red y mostrar información sobre ellas.
- nmap: Realiza un mapeo de puertos a cualquier servidor, para determinar que sistema operativo esta utilizando, o que puertos tiene abiertos.
- ping: Permite enviar paquetes de información a través de la red para determinar si se tiene acceso a una máquina remota.
- traceroute: Permite trazar una ruta de hosts por donde pasa un paquete para llegar al host destino seleccionado.

Bitácoras del sistema.

Un administrador de una máquina Linux, revisa los archivos de registro, ya que éstos sirven para detectar posibles fallos de seguridad en la máquina Linux. Los archivos de registro se encuentran en /var/log los cuales son los siguientes:

- messages: Aquí se guardan los mensajes de seguridad y autenticación.
- cron: Mensajes generados por el demonio cron.
- secure: Mensajes de seguridad de acceso (conexiones, wrappers).
- access_log: Mensajes de acceso al servidor WWW, si lo hubiese.
- error_log: Mensajes de error durante el acceso al servidor WWW, si lo hubiese.
- utmp: Contiene información sobre los usuarios que se encuentran. Actualmente en el sistema. La información se consulta con comandos como *who, w, users*.
- wtmp: Contiene información sobre las entradas y salidas de los usuarios al sistema. La información se consulta con el comando *last*.

1.6.-ADMINISTRACIÓN DE BASE DE DATOS.

1.6.1 Tareas del Administrador de Base de Datos.

Existen diversas tareas que el administrador de bases de datos debe realizar, entre las que destacan:

- Monitoreo del sistema manejador de base de datos.
- Respaldos de las bases de datos y restauración de las mismas.
- Asignación y revocación de privilegios.
- Verificar el rendimiento del DBMS.
- Actualización del sistema.
- Administración de dispositivos.
- Instalación del servidor.
- Implementar y cuidar que se cumplan las políticas de seguridad de las BD.
- Crear las bases de datos.
- Creación de usuarios.
- Verificar que se este cumpliendo la integridad de las bases de datos.

1.6.2 Scripts.

El administrador de base de datos, debe ser capaz de implementar diversos scripts[30] que le ayuden en la tarea de monitoreo del sistema de base de datos, ya que por ejemplo debe generar un plan de respaldos de los datos que existen en las bases, ya que estos son el activo más preciado para las empresas.

Se puede ayudar ya sea de herramientas mismas de la base de datos, o bien realizando ciertos programas que se ejecuten a cierta hora determinada del dia, con esto el administrador garantizara que la tarea que debe de realizar se realice constantemente y así poder cumplir con sus funciones de administración de una manera más eficiente.

Por lo tanto el administrador de base de datos, debe ser capaz de implementar diversos scripts que le ayuden a automatizar sus labores.

1.6.3 Administrador de Base de Datos en Sybase.

Base de datos del sistema.

Al instalar el sistema manejador de bases de datos Sybase se crean automáticamente estas bases de datos del sistema:

- La base de datos *master*
- La base de datos *model*
- La base de datos de procedimientos del sistema, *sybsystemprocs*
- La base de datos temporal, *tempdb*

Optativamente, es posible instalar:

- La base de datos de auditoría, *sybsecurity*.
- El ejemplo de base de datos, *pubs2.*
- La base de datos de sintaxis, *sybsyntax.*

[30] Un script es un programa el cual se ejecuta a determinada hora para realizar una tarea en especifico.

Las bases de datos *master*, *model* y *tempdb* residen en el dispositivo indicado durante la instalación, que se conoce como *master*. La base de datos *master* está contenida por completo en el dispositivo master y no puede expandirse en ningún otro. Las demás bases de datos y objetos de usuario deben crearse en otros dispositivos.

No se deben de almacenar bases de datos de usuario en el dispositivo master, pues dificultaría la recuperación de las bases de datos del sistema en caso de que resultaran dañadas. Además, es posible que no pueda recuperar bases de datos de usuario guardadas en *master.*

La base de datos *sybsecurity* debe instalarse en su propio dispositivo y segmento.

Sybsystemprocs puede instalarse en un dispositivo cualquiera, pero por default se instala en el dispositivo sysprocsdev.

Base de datos master.

La base de datos *master* controla el funcionamiento de Sybase en su conjunto y almacena información sobre todas las bases de datos de usuario, y sus dispositivos asociados. Hace el seguimiento de:

- Cuentas de usuarios (en *syslogins*)
- Cuentas de usuarios remotos (en *sysremotelogins*)
- Servidores remotos con los que puede interactuar este servidor (en *sysservers*)
- Procesos en curso (en *sysprocesses*)
- Variables de entorno configurables (en *sysconfigures*)
- Mensajes de error del sistema (en *sysmessages*)
- Bases de datos en SQL Server (en *sysdatabases*)
- Espacio asignado a cada base de datos (en *sysusages*)
- Cintas y discos montados en el sistema (en *sysdevices*)
- Bloqueos activos (en *syslocks*)
- Juegos de caracteres (en *syscharsets*) e idiomas (en *syslanguages*)
- Usuarios que tienen roles en todo el servidor (en *sysloginroles*)
- Roles del servidor (en *syssrvroles*)
- Máquinas Sybase que están en línea (en *sysengines*)

Dado que *master* guarda información sobre los dispositivos y bases de datos de usuario, es necesario estar en la base de datos *master* para poder ejecutar create database, alter database, disk init, disk refit, disk reinit y los comandos de duplicación de disco.

Base de datos model.

Sybase incluye la base de datos *model,* que sirve de plantilla, o prototipo, de las nuevas bases de datos de usuario. Cada vez que un usuario introduce el comando create database, el manejador de base de datos realiza una copia de la base de datos *model* y extiende la copia al tamaño especificado por el comando create database. Una base de datos nueva no puede ser más pequeña que model.

La base de datos *model* contiene las tablas del sistema necesarias para cada base de datos de usuario. Se puede modificar *model* para personalizar la estructura de las nuevas bases de datos creadas. Todo lo que haga en *model* se reflejará en cada base de datos nueva. Algunos de los cambios que los administradores suelen efectuar son:

- Añadir reglas, valores predeterminados o tipos de datos definidos por el usuario.
- Añadir los usuarios que deben tener acceso a todas las bases de datos de SQL Server.
- Conceder privilegios predeterminados, en especial para las cuentas huéspedes.
- Definir opciones de base de datos, como select into/bulkcopy.

Normalmente, la mayoría de los usuarios no tienen permiso para modificar la base de datos *model.* Tampoco tiene sentido conceder permiso de lectura, puesto que SQL Server copia todo su contenido a cada nueva base de datos de usuario.

El tamaño de *model* no puede ser mayor que el de *tempdb*. Sybase muestra un mensaje de error si se intenta aumentar el tamaño de *model* sin dar a *tempdb* ese mismo tamaño como mínimo.

Base de datos sybsystemprocs.

Los procedimientos del sistema de Sybase se almacenan en la base de datos *sybsystemprocs.* Cuando un usuario ejecuta, desde cualquier base de datos, un procedimiento almacenado cuyo nombre comienza con "sp_", Sybase lo busca primero en la base de datos actual del usuario. Si no está ahí, lo busca en *sybsystemprocs*. Si tampoco está, Sybase busca el procedimiento en *master.*

Si el procedimiento modifica tablas del sistema (por ejemplo, sp_adduser modifica la tabla *sysusers),* los cambios se realizan en la base de datos desde la que se ejecutó el procedimiento.

Si desea cambiar los permisos predeterminados sobre los procedimientos del sistema, debe hacerlo en *sybsystemprocs.*

Base de datos tempdb.

Sybase tiene una base de datos temporal, *tempdb*, que proporciona un área de almacenamiento para tablas temporales y otras necesidades temporales (por ejemplo, resultados intermedios de group by y order by). El espacio de *tempdb* se comparte entre todos los usuarios de todas las bases de datos del servidor.

El tamaño predeterminado de *tempdb* es 2MB. Ciertas actividades pueden requerir el incremento de su tamaño.

Las más comunes son:

- Tablas temporales grandes.
- Mucha actividad en tablas temporales, que llena los diarios *tempdb.*
- Ordenaciones grandes o muchas ordenaciones simultáneas. Las subconsultas y agregados con group by también causan cierta actividad en *tempdb*.

Se puede aumentar el tamaño de *tempdb* con el comando *ALTER DATABASE*. La base de datos *tempdb* se crea inicialmente en el dispositivo master. Puede añadirse espacio del master o de cualquier otro dispositivo de base de datos.

Base de datos sybsecurity.

La base *sybsecurity* contiene el sistema auditor de SQL Server y consta de:

- La tabla *sysaudits,* que contiene la lista de auditoría. Todos los registros de auditoría se escriben en *sysaudits.*
- La tabla *sysauditoptions,* que contiene filas que describen las opciones globales de auditoría.
- Las demás tablas predeterminadas del sistema que se derivan de *model.*

Base de datos pubs2.

Diseñada como utilidad de aprendizaje, *pubs2* es la base de la mayoría de los ejemplos presentados en la documentación de Sybase.

La base de datos de ejemplo incluye un mecanismo de usuario invitado que permite el acceso a la base de datos a cualquier Sybase autorizado. Este usuario tiene amplios privilegios en *pubs2,* incluyendo permisos para seleccionar, insertar, actualizar y eliminar tablas de usuario.

La base de datos *pubs2* requiere por lo menos 2MB.

Base de datos sybsyntax.

La base de datos *sybsyntax* contiene ayuda sobre la sintaxis de los comandos de Transact-SQL, los procedimientos del sistema de Sybase, las utilidades y las rutinas. Los usuarios pueden recuperar esa información mediante el procedimiento del sistema *sp_syntax.*

Por ejemplo, para aprender la sintaxis del comando select de Transact-SQL, se escribe:

```
sp_syntax "select"
```

Tablas de sistema.

La base de datos master contiene tablas del sistema que controlan la información sobre Sybase de forma global. Además, cada base de datos (incluida *master)* contiene tablas del sistema que controlan información específica de esa base de datos.

Todas las tablas suministradas con Sybase en la base de datos *master* se consideran tablas del sistema, las cuales también pueden denominarse diccionario de datos o catálogos del sistema. Además, cada base de datos de usuario se crea con un subconjunto de estas tablas del sistema.

Cuando se instala Sybase, se crean una base de datos *master* y sus tablas. Las tablas del sistema de una base de datos de usuario se crean automáticamente cuando se ejecuta el comando create database. Los nombres de todas las tablas del sistema comienzan con "*sys*".

Consulta de las tablas del sistema.

Es posible realizar consultas en las tablas del sistema igual que en cualquier otra tabla. Por ejemplo, la siguiente instrucción devuelve los nombres de todos los disparadores de la base de datos:

```
SELECT name
FROM sysobjects
WHERE type = "TR"
```

Además, Sybase cuenta con procedimientos almacenados (llamados *procedimientos del sistema*), muchos de los cuales proporcionan atajos para consultar las tablas del sistema.

Estos procedimientos del sistema proporcionan información de las tablas del sistema como se puede apreciar en la tabla 1.13.

Las claves primarias, externas y comunes de las tablas del sistema se definen en las bases de datos *master* y *model.* Para obtener un informe sobre las claves definidas, se debe ejecutar el procedimiento del sistema *sp_helpkey*. Para un informe sobre las columnas de dos tablas del sistema que pueden ser susceptibles de combinación, se deberá ejecutar *sp_helpjoins*.

sp_commonkey	sp_helpindex	sp_helpsort
sp_configure	sp_helpjoins	sp_helptext
sp_dboption	sp_helpkey	sp_helpthreshold
sp_estspace	sp_helplanguage	sp_helpuser
sp_help	sp_helplog	sp_lock
sp_helpconstraint	sp_helpremotelogin	sp_monitor
sp_helpdb	sp_helprotect	sp_spaceused
sp_helpdevice	sp_helpsegment	sp_who
sp_helpgroup	sp_helpserver	

Tabla 1.13.- Procedimientos almacenados del sistema.

Procedimientos del sistema.

Los nombres de todos los procedimientos del sistema comienzan con "*sp_*", y están situados en la base de datos *sybsystemprocs*, pero muchos pueden ejecutarse desde cualquier base de datos.

Excepto por los procedimientos del sistema que actualizan sólo las tablas de *master*, si se ejecuta un procedimiento del sistema sobre una base de datos que no sea *sybsystemprocs*, funciona sobre las tablas del sistema de la base de datos desde la que se ejecutó. Por ejemplo, si el dueño de la base de datos *pubs2* ejecuta *sp_adduser* desde *pubs2,* el nuevo usuario se añade en *pubs2.sysusers.*

Uso de los procedimientos del sistema

Si el valor de un parámetro de un procedimiento del sistema contiene palabras reservadas, signos de puntuación o espacios en blanco incrustados, debe ir entre comillas simples o dobles. Si el parámetro es el nombre de un objeto y se califica mediante el nombre de su base de datos o de su propietario, todo el nombre debe ir entre comillas.

Las sesiones pueden invocar procedimientos del sistema mediante modos encadenados o no encadenados de transacción. Sin embargo, los procedimientos del sistema que modifican datos en las tablas del sistema de *master* no pueden ejecutarse desde dentro de una transacción, ya que esto podría comprometer su recuperación. Los procedimientos del sistema que crean tablas temporales no pueden ejecutarse desde transacciones.

Los administradores del sistema pueden escribir procedimientos ejecutables desde cualquier base de datos. Basta crear un procedimiento almacenado en *sybsystemprocs* y darle un nombre que comience con "sp_". El *uid* del procedimiento almacenado debe ser 1, que es el *uid* del dueño de la base de datos.

Configuración e instalación del servidor.

Para la instalación del sistema manejador de base de datos Sybase lo primero que se debe hacer es bajar algunos archivos que son esenciales para la instalación de éste, los cuales son: sybase-common-11.9, sybase-ase-11.9, sybase-spanish-11.9 (este es opcional ya que sirve para dar soporte para idioma español), sqsh-2.1-linux-11.9 y el archivo sqshrc.txt.

Ahora una vez que se cuenta con esos archivos se prosigue con la instalación, y se ejecutan los siguientes comandos:

```
# rpm –ivh sybase-common-11.9
# rpm –ivh sybase-ase-11.9
# rpm –ivh sybase-spanish-11.9
```

Estas instrucciones se utilizan para la instalación del servidor, ahora sólo falta configurarlo para que se pueda comenzar a utilizar, el comando rpm es un comando para instalar paquetes rpm´s en sistema Linux (Red Hat).

Después lo que se tiene que hacer es cambiarle la contraseña al usuario sybase el cual se crea automáticamente al instalar el servidor, y esto se hace con el siguiente comando:

```
#passwd sybase
```

Ahora con los rpm´s se crea un directorio en /opt/sybase-11.9/ al cual se ingresa.

```
#cd /opt/sybase-11.9
```

Se cambia de dueño a todos los archivos que están en el directorio para que sean propiedad de Sybase.

```
#chown –R sybase:sybase * .profile .cshrc
```

Ahora se debe de renombrar el archivo que ya se tiene listo para la instalación el cual es sqshrc.txt y se le renombra como .sqshrc en el home de root y en opt/sybase/11.9

Enseguida se instala el archivo sqsh-2.1-linux con los siguientes comandos.

```
#tar –zxvf sqsh-2.1-linux
#cd sqsh-2.1
#cp sqsh /etc/
#cp sqsh /usr/local/bin
```

Lo anterior se hace para instalar el cliente de sybase que va a ayudar a conectar al sistema manejador de base de datos el cual es más parecido al shell de linux y evita interactuar con el servidor de base de datos con el cliente que trae por default es cual es isql, ya que éste es un poco más limitado.

Ahora hay que configurar el servidor de base de datos, para ello se termina la sesión de root y se loguea con el usuario sybase y se entra al directorio bin.

Aquí en este punto se ejecuta el comando *srvbuild* y aparece una pantalla como la que se muestra en la fig. 1.16 en la cual vamos a empezar a configurar nuestro servidor.

En este punto se permite escoger si se quiere instalar un servidor de base de datos o un servidor de respaldos, una vez que se escoge instalar el servidor de base de datos se asigna un nombre para poder identificarlo y se le da clic en el botón OK.

Ahora aparece otra pantalla como la que se muestra en la fig. 1.17 en la cual se va a configurar en donde se encontraran los dispositivos master y sybsystemprocs además de especificar el tamaño que se le va a asignar para éstos.

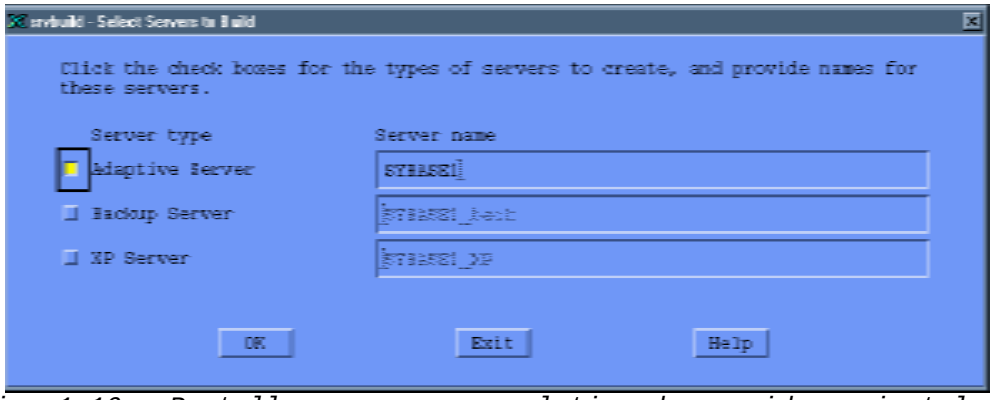


Fig. 1.16.- Pantalla para escoger el tipo de servidor a instalar.

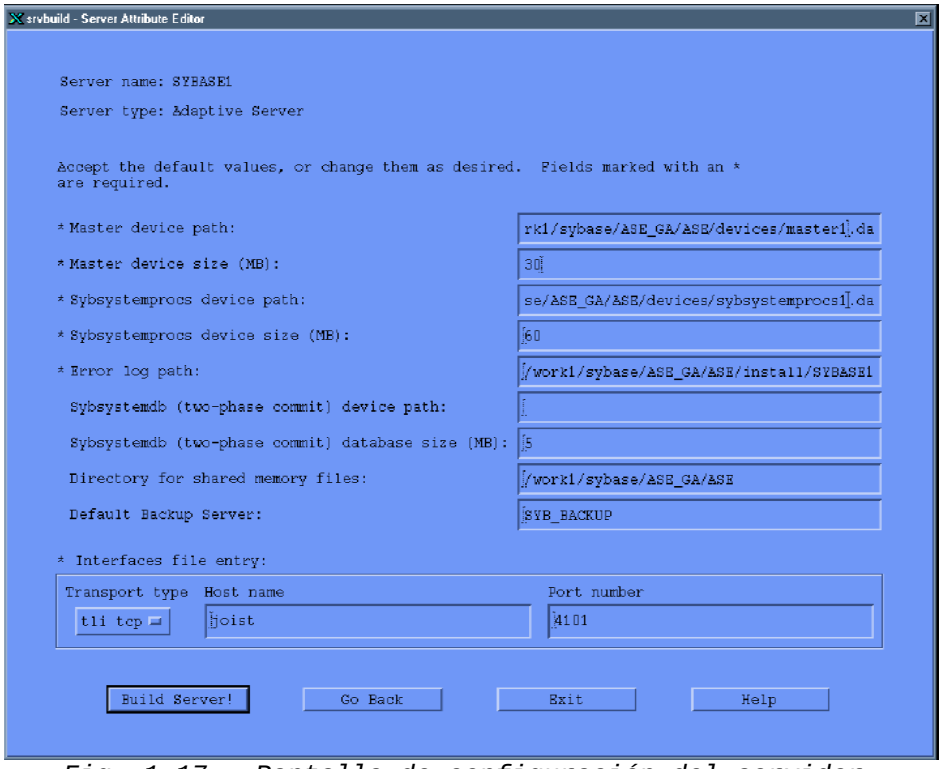


Fig. 1.17.- Pantalla de configuración del servidor.

Una vez que se termina de instalar el servidor, automáticamente en el home de sybase se crean varios archivos, los importantes para nosotros son SQL.cfg e interfaces, en estos archivos se puede cambiar la configuración de sybase.

También se crean unos archivos llamados RUN_SQL y RUN_SQL_back los cuales facilitan el levantar el servidor de base de datos y el de respaldos.

Así con estos archivos será más fácil el iniciar y parar el servidor.

Inicialización y terminación del servidor.

Para levantar el servidor se ejecuta el siguiente comando el cual ayudara a inicializar el servidor.

```
# /opt/sybase-11.9/install/startserver –f RUN_SQL
```

En la ventana que levanta el servidor se deja abierta y se abre otra para que ahí se pueda conectar al servidor, ya que en la ventana donde se levanto el servidor envía mensajes de lo que se esta haciendo, así como los errores que se presentan en la ejecución de los consultas.

Ahora ya se puede conectar al servidor y se hace de la siguiente manera:

#sqsh –Usa -SSQL

Con lo anterior cambia el promt. De la pantalla por el símbolo “>”, con esto se sabe que ya se esta dentro del servidor y se puede empezar a utilizarlo con sentencias SQL.

Ahora para dar de baja el servidor existen 3 formas de hacerlo las cuales son:

1. Esta es la mejor opción, ya que con este comando espera a que terminen todas las transacciones y después se detiene el servidor, esto lo hace para que no exista perdida de información si se esta trabajando con los datos.

```
> shutdown;
```

2. Con este segundo comando no espera a que terminen los procesos o transacciones, así que puede haber pérdida de datos.

```
> shutdown with nowait;
```

3. Esta tercera forma no es muy recomendable para dar de baja el servidor porque se pueden dañar las bases de datos, y este comando se realiza desde fuera del servidor de base de datos, es decir, desde el sistema operativo.

```
# kill -9 PID
```

Archivos de interfaces.

Sybase puede comunicarse con otros servidores Sybase’s, aplicaciones Open Server y programas cliente en la red. Los clientes pueden hablar con uno o más servidores, y los servidores pueden comunicarse entre sí

mediante llamadas de procedimientos remotos. Para que los productos interactúen unos con otros, cada uno de ellos debe saber dónde residen los otros en la red. Esta información se almacena en un archivo de interfaces (que puede llamarse *interfaces, interfaz* o *sql.ini,* dependiendo del sistema operativo).

El archivo de interfaces es como una agenda de direcciones. Contiene el nombre y dirección de todos los servidores conocidos. Cuando se emplea un programa cliente para conectarse a un servidor, el programa busca el nombre del servidor en el archivo de interfaces y entonces se conecta al servidor usando la dirección.

Aquí también en este archivo esta contenido el puerto por el cual escucha las peticiones el servidor de base de datos.

Administración de los recursos físicos.

Sybase puede tomar algunas decisiones predeterminadas razonables sobre muchos aspectos del manejo del almacenamiento, como dónde situar las bases de datos, tablas e índices y cuánto espacio se asigna a cada uno. No obstante, el administrador del sistema tiene el control final sobre la asignación de recursos de disco al servidor de base de datos y la ubicación física de las bases de datos, tablas e índices en esos recursos.

Un dispositivo es un espacio en disco que se reserva, además a ese espacio se le da formato sybase. Los dispositivos se crean en tamaños por páginas. Una página es de 2K, entonces para crear un dispositivo se multiplica el tamaño de MB por 512 para obtener el tamaño en páginas, ya que cuando se crea un dispositivo para alojar a las bases de datos, el tamaño se debe poner en páginas.

El comando que se utiliza para crear los dispositivos es:

```
>disk init
       name=”nombre del dispositivo”
       physname=”ruta donde se va a guardar el dispositivo”
       vdevno=”numero virtual del dispositivo”
       size=”tamaño del dispositivo en páginas”
       go
```

Sólo los administradores del sistema pueden usar disk init. Antes de ejecutar el comando hay que considerar los siguientes puntos:

- Respaldar la base de datos *master*
- Asegurarse de que hay suficiente espacio en disco
- Asegurarse de que el archivo o dispositivo no haya sido ya inicializado
- Asegurarse que la cuenta *sybase* tenga permisos de escritura sobre ese dispositivo
- Máximo de dispositivos de base de datos configurables: 255

El administrador del sistema debe tomar muchas decisiones respecto a la asignación física de espacio a las bases de datos de Sybase. Las consideraciones principales son:

- *Recuperación:* La duplicación de discos y/o el mantenimiento de diarios en un dispositivo físico distinto suponen dos mecanismos para la recuperación total en caso de fallos físicos de los discos.

- *Rendimiento:* Para ciertas tablas o bases de datos en que la velocidad de las lecturas y escrituras en disco es crucial, la ubicación correcta de los objetos de base de datos en dispositivos físicos produce mejoras en el rendimiento. La duplicación de discos reduce la velocidad de las escrituras en disco.

Tablas del sistema que manejan el almacenamiento

Dos tablas del sistema de la base de datos *master* y otras dos de cada base de datos de usuario controlan la ubicación de las bases de datos.

Tabla sysdevices

La tabla *sysdevices* de la base de datos *master* contiene una fila para cada dispositivo de base de datos y puede contener una fila para cada dispositivo de volcado (cinta, disco o archivo del sistema operativo) disponible para el servidor de base de datos.

El comando *disk init* añade entradas de dispositivos de base de datos a *master.sysdevices*. La tabla *sysdevices* almacena dos nombres por cada dispositivo.

Un nombre lógico o de dispositivo, usado en los sucesivos comandos de administración del almacenamiento, se guarda en la columna *name* de *sysdevices.* Generalmente es un nombre fácil de asociar que indica el uso previsto del dispositivo, por ejemplo "dipositivo1".

El nombre físico es el nombre real del dispositivo en el sistema operativo. Este nombre se utiliza exclusivamente en el comando disk init; después, todos los comandos de almacenamiento de datos emplean el nombre lógico.

Una base de datos puede residir en uno o más dispositivos, y un dispositivo puede almacenar una o más bases de datos.

Tabla sysusages

La tabla *sysusages* de la base de datos *master* lleva cuenta de todo el espacio asignado a todas las bases de datos. Los comandos create database y alter database asignan nuevo espacio a la base de datos añadiendo una fila a *sysusages* para cada dispositivo de base de datos o fragmento de dispositivo.

Los procedimientos del sistema sp_addsegment, sp_dropsegment y sp_extendsegment cambian la columna *segmap* de *sysusages* por el dispositivo que se correlaciona o no se correlaciona a un segmento.

Tabla syssegments

La tabla *syssegments* (una en cada base de datos) enumera los segmentos de una base de datos. Un segmento es un conjunto de dispositivos de base de datos y/o fragmentos disponibles para una base de datos en particular. Las tablas e índices pueden asignarse a cierto segmento y, por lo tanto, a cierto dispositivo físico, o pueden abarcar varios dispositivos físicos.

Tabla sysindexes

La tabla *sysindexes* enumera cada tabla e índice, y el segmento en que se almacena cada tabla, índice y cadena de páginas de texto. También contiene otra información, como la configuración de max_rows_per_page para la tabla o índice.

Usuarios del sistema y control de acceso.

Un servidor de BD debe tener seguridad de tres niveles los cuales son:

1. Seguridad al entrar al servidor, es decir, debe ser un usuario válido en el servidor de BD.

2. Debe ser un usuario válido en la base de datos.

3. Debe tener permisos sobre los objetos de la base de datos.

Un usuario que es valido en el servidor de base de datos es llamado login.

Para crear un login se utiliza el siguiente procedimiento almacenado "*sp_addlogin*":

```
> sp_addlogin "nombre del usuario","password","con que base de datos se conecta por default","idioma con el que va a interactuar","a quien esta asociado el login";
```

Existen 3 roles de login's en el sistema manejador de base de datos Sybase los cuales son:

- Rol SA (Administrador de sistema)
- Rol SSO (Oficial de seguridad del sistema)
- Rol OPER (Operadores)

Rol sa

El rol sa es análogo al administrador del sistema para sybase y puede realizar las siguientes funciones:

- Manejo del almacenamiento del disco.
- Borrar, modificar, bloquear, y desbloquear logines.
- Otorgar/revocar roles sa.
- Crear bases de datos de usuario.
- Otorgar ciertos permisos a los usuarios de Sybase.
- Afinar el servidor de base de datos cambiando los parámetros de configuración.
- Cerrar el servidor de base de datos y sus procesos.
- Monitorear la recuperación de base de datos en el arranque del servidor y utilizar ciertas herramientas para el diagnóstico de problemas en el sistema.

Rol SSO

Este tipo de rol es llamado el oficial de seguridad del sistema, ya que realiza las siguientes actividades:

- Crean logines en el servidor de BD (asignando passwords iníciales).
- Cambian passwords.
- Fijan un intervalo de expiración del password.
- Crean, otorgan, y revocan roles de usuario.
- Otorgan autorización para uso del proxy[31].

[31] Proxy es un software que permite a varias computadoras acceder a Internet a través de una única conexión física.

- Otorgan y revocan roles SSO y OPER.
- Manejan el sistema de auditoría.
- Bloquean y desbloquean logines.

Los Oficiales de Seguridad del Sistema: SSO no puede modificar o borrar logines; esto sólo puede hacerlo un login con rol sa.

Rol OPER

Este tipo de rol se dedica a hacer lo siguiente:

- Los Operadores pueden respaldar y cargar todas las bases de datos y logs de transacciones en el servidor.
- Realizan dump db, dump transaction, load db, y load transaction.
- No tienen que ser propietarios de una base de datos para realizar tareas de mantenimiento.
- Los propietarios de base de datos realizan tareas de mantenimiento de sus propias bases de datos en lugar de, o en adición a, el operador.

Para crear un rol de estos tipos anteriores basta con utilizar el procedimiento almacenado "*sp_role*" y se hace de la siguiente forma:

```
> sp_role "grant","sa_role","login1";
> sp_role "grant","sso_role","login2";
> sp_role "grant","oper_role","login3";
```

Para borrar un login se utiliza el procedimiento almacenado "*sp_droplogin*"

```
> sp_droplogin login1;
```

Pero para poder borrar un login no debe ser usuario dentro de una base de datos, o ser dueño de una base de datos.

DBO.

El dbo es el propietario de la base de datos y este puede realizar las siguientes actividades:

- Tiene la autorización de crear bases de datos.
- Adiciona y retira usuarios de la base de datos con sp_adduser.
- Otorga y revoca permisos a usuarios para crear objetos en la base de datos y ejecutar comandos con *GRANT.*

Para agregar un usuario a una base se datos se hace con el siguiente procedimiento almacenado, pero antes se debe estar en la base de datos donde se quiere agregar al usuario, para eso se utiliza *use "nombre de la base"*.

```
> sp_adduser usuario2;
```

Análogamente para borra un usuario se hace de la siguiente manera:

```
> sp_dropuser usuario2;
```

En este caso si el usuario que se quiere borrar es dueño de la base de datos no se podrá, hasta que se asigne un nuevo dueño a la base de datos o desaparezca esa base de datos. Para cambiar de dueño la base de datos se realiza de la siguiente forma:

```
> sp_changedbowner usuario3;
```

La otra tarea que puede realizar un dbo es la de dar o revocar permisos sobre los objetos de las bases de datos, para esto se auxilian del comando grant y revoke respectivamente.

```
> grant [permisos] on [nombre_objeto] to [usuario];
```

```
> grant select on tabla1 to usuario3;
```

Y para quitar permisos se utiliza:

```
> revoke [permisos] on [nombre_objeto] from [usuario];
```

```
> revoke update on tabla1 from usuario4;
```

Creación de Base de Datos.

En la creación de una base de datos se tienen que tomar en cuenta muchos factores, los cuales determinan son el tamaño, ubicación, si lleva un dispositivo para log[32], etc.

El comando que se utiliza es el siguiente, aunque tiene ciertas variantes en su utilización.

```
> CREATE DATABASE db1;
> CREATE DATABASE db2 on dev_dat1=2;
> CREATE DATABASE db3 on dev_dat1=2 log on dev_log=1;
```

El primer comando creara una base de datos llamada db1, pero como no se le puso de que tamaño seria toma por default el tamaño de la base de datos *model*, ya que esta base de datos es el modelo para crear nuevas bases de datos.

En el segundo caso el comando creará una BD llamada db2 la cual tendrá un tamaño de 2MB y estará alojada en el dispositivo dev_dat1. Y en el tercer caso se creará una base de datos db3 con un tamaño de 2MB para datos y 1MB para log.

El log de transacción es un espacio que se reserva para que se realicen las transacciones que se realizan cuando se consultan, insertan, borran o actualizan los datos de la base de datos.

Para calcular el tamaño del log depende de que tipo de sistema de base de datos se crea ya que si es un sistema DSS, es decir, que sólo por lo regular tendrá consultas, el valor del log debe ser del 15% del tamaño de los datos, mientras que si es un servidor de base de datos OLTP en el cual se van hacer mas inserciones que consultas, entonces el tamaño del log es 30% del tamaño de los datos.

[32] Un log es un registro de eventos para saber quien, que, cuando, donde y porque de un evento que ocurre en nuestro sistema.

Cuando se crea una base de datos, hay que tratar de crearla con el suficiente espacio para que se almacenen los datos de una forma adecuada, por lo que existe una técnica la cual ayuda a calcular manualmente el tamaño de la base de datos, los pasos a seguir para el calculo es:

1. Crear la estructura de las tablas (No todas sólo las que crecerán mucho).
2. Una vez creadas verificar el tamaño del registro considerando el tipo de dato de cada columna según la plataforma.
3. Saber el número de registros en el punto 0, es decir, que con cuantos registros va a empezar la base de datos.
4. Ya que se sabe esto se multiplica el tamaño del registro por el número de registros.
5. Una vez que se hizo lo anterior el resultado se multiplica por 35% y ese resultado se le suma al resultado anterior, con lo cual dará el tamaño de la base de datos.
6. Por último calcular el tamaño del log dependiendo de que tipo de sistema que se tenga y se le suma esa cantidad al total que antes se había obtenido.

Esta técnica se utiliza para calcular el tamaño de la base de datos de forma manual, pero Sybase provee de un procedimiento almacenado que simplifica esta tarea y su sintaxis es la siguiente:

```
> spestspace nombre_tabla, numero_de_registros
> spestspace tabla1, 300,000
```

Y esto da el tamaño para los datos, nos faltaría definir solo el tamaño para el log y sumarle este al tamaño para los datos.

Hay veces en que se quiere aumentar el tamaño de la base de datos en otro dispositivo porque ya no se tiene espacio en el que esta creada, y se realiza ejecutando el siguiente comando:

```
> alter database db3 on dev_dat2=5 log on dev_log1=1;
```

Bitácora de errores.

Existe un archivo donde el servidor de base de datos envía los mensajes de error que ha generado, debido a una falla en él ya sea de consultas o porque no esta funcionando correctamente.

Usualmente esta localizado en el directorio install en el directorio de Sybase, por lo general este archivo se llama SQL.log y se tiene que estar revisando continuamente.

Realización de respaldos y restauraciones.

Un respaldo es el copiado ya sea de estructura, datos o ambos del servidor, para tratar de tener guardados los datos por si se sufre cualquier contratiempo en el servidor de base de datos y así poder restaurar los datos para que no se tengan perdida de la información, ya que ésta es lo más preciado para la institución.

Lo primero que se debe verificar es que se tenga instalado el servidor de backup[33] y verificar que haya comunicación entre el servidor de base de datos y el backup's.

El comando para realizar un respaldo es el siguiente:

[33] Un servidor de backup es una maquina en la cual su función principal es la guardar o realizar respaldos.

```
> dump database db3 to “/opt/sybase-11.9/dev/db3.dbf”;
```

Pero el comando anterior sólo sirve para respaldar la parte de los datos, pero si se quiere respaldar el log se realiza de la siguiente forma:

```
> dump transaction db3 to “/opt/sybase-11.9/dev/db3.log1”
```

Con el comando anterior se hace el respaldo de log además de que se hace limpieza del log de transacciones.

Una cosa importante que se debe considerar es que al restaurar la base de datos se debe tener el mismo tamaño de la base de datos y del dispositivo donde estaba creada.

Para restaurar una base de datos se debe primero crear la base de datos y después se realiza la restauración con el siguiente comando:

```
> load database db3 from “/opt/sbase-11.9/dev/db3.dbf”;
```

Y para restaurar el log se hace de la siguiente manera:

```
> load tran db3 from “/opt/sybase-11.9/dev/db3.log1”;
```

El respaldo de lo datos es vital, por lo que se debe tener un mecanismo que haga los respaldos automáticamente y a una hora determinada para que se tenga copia de los datos, por si ocurre alguna falla.

Monitoreo de la consistencia de la base de datos.

El monitoreo del sistema de base de datos es muy importante, ya que éste provee de un panorama general de como está funcionando el servidor de base de datos, además de que puede ayudar a reconocer errores que pueden estar sucediendo y tratar de resolver esos errores.

Lo primero que se debe verificar es el archivo de error log, ya que ahí se encuentran toda la información de los errores que se han estado presentando en el servidor de BD.

Otra cosa que se debe monitorear constantemente es el espacio de almacenamiento. Otros comandos que pueden ayudar al monitoreo son:

```
> dbcc checkalloc (“db3”);
> dbcc checkcatalog (“db3”);
> dbcc checkalloc (“db3”,”fix”);
> dbcc dbrepair(“db3”,”dropdb”);
> dbcc reindex (“tabla1”);
```

El primer comando sirve para hacer un chequeo de página por página del alojamiento de los registros de todas las tablas.

El siguiente comando ayuda a checar todas las tablas del sistema, es decir las que tienen prefijo *sys* para verificar que estén funcionando correctamente.

El tercer comando es igual al primer comando, sólo que tiene un argumento más, el cual sirve para arreglar la base de datos de errores menores.

El siguiente comando se utiliza cuando la base de datos se corrompe y ya no se puede borrar la base de datos con *drop database*.

El último comando ayuda a regenerar los índices de cada tabla, pero esto sólo se hace cuando se cambia el set de ordenamiento[34].

1.6.4. Administración de Bases de Datos en Postgresql

Instalación y configuración de Postgresql.

Lo primero que se debe hacer es bajar el paquete de postgresql. Una vez que se cuenta con el paquete se ejecutan las siguientes líneas:

```
# groupadd postgres
# useradd -g postgres postgres
# tar -zxvf /home/instalacion/postgresql-8.1.1.tar.gz
# ln -s postgresql-8.1.1 postgresql
# cd postgresql
# ./configure --prefix=/usr/local/pgsql
# make
# make install
# mkdir /usr/local/pgsql/data
# chown postgres /usr/local/pgsql/data
# su - postgres
# PATH=$PATH:/usr/local/pgsql/bin
# export POSTGRES_HOME=/usr/local/pgsql
# export PGDATA=/usr/local/pgsql/data
# export PGLIB=/usr/local/pgsql/lib
# export LD_LIBRARY_PATH=/usr/local/pgsql/lib
# /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
```

Formas de levantar PostgreSQL

```
# /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
#/usr/local/pgsql/bin/pg_ctl      –l      /usr/local/pgsql/data/errors.log      -D
/usr/local/pgsql/data/ -o –i start
# /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
# /usr/local/pgsql/bin/psql template1
# exit
# cp /usr/local/pgsql/bin/postmaster /etc/rc.d/init.d/postgres
# ln -s /etc/rc.d/init.d/postgres /etc/rc.d/rc5.d/S98postgres
```

[34] Un set de ordenamiento es aquel que se utiliza para saber como esta ordenando los datos el sistema de gestión de la base de datos.

Configuración

El primer paso para el procedimiento de instalación es configurar el árbol de fuentes para el sistema donde se va a instalar y elegir las opciones que se requieran. Esto se hace mediante el script *configure*. Para tomar los valores por default:

```
# ./configure
```

Este script ejecutará ciertas pruebas para obtener los valores de determinados sistemas y detectar algunas especificaciones del sistema operativo para finalmente crear archivos para registrar lo que encontró.

La configuración por default construirá el servidor y las utilidades, así como las interfaces y aplicaciones del cliente. Todos los archivos serán instalados bajo el directorio /usr/local/pgsql.

Para modificar el directorio de instalación se utiliza la opción prefix, donde PREFIX es la ruta hacia el directorio de instalación.

```
--prefix=PREFIX
```

Construcción

La construcción puede tomar desde cinco minutos hasta una hora y media dependiendo del hardware, para comenzar la construcción, se teclea

```
# make
```

Instalación de los archivos

Si se esta actualizando un sistema existente y se van a instalar nuevos archivos sobre los viejos, entonces se deberían respaldar los datos y apagar el viejo servidor, para instalar PostgreSQL tecleamos:

```
# make install
```

La instrucción anterior instalará los archivos en los directorios que fueron especificados durante la configuración.

Después de la instalación es importante modificar el archivo /etc/profile con la siguiente información.

```
PATH=/usr/local/pgsql/bin:$PATH
MANPATH=/usr/local/pgsql/man:$MANPATH
PGDATA=/usr/local/pgsql/data
LD_LIBRARY_PATH=/usr/local/pgsql/lib
export PATH MANPATH PGDATA LD_LIBRARY_PATH
```

Manejo del proceso del servidor Postgresql.

Como ya se ha visto existen varias formas de iniciar el servicio de postgresql, entre las cuales se pueden encontrar las siguientes:

```
1. # /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
2. #/usr/local/pgsql/bin/pg_ctl   –l   /usr/local/pgsql/data/errors.log   -D
   /usr/local/pgsql/data/ -o –i start
3. # /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data &
```

Postmaster es el daemon[35] en si, por lo que al lanzarlo directamente, hará un fork[36] y se pondrá en background[37] a escuchar las peticiones de los clientes y usuarios que se conecten a las bases de datos.
Una vez que se inicializa la base de datos, hay que activar el servicio postmaster para aceptar conexiones. Si se desea aceptar las conexiones utilizando el protocolo TCP/IP, se debe ejecutar la siguiente instrucción:

```
/usr/local/pgsql/bin/postmaster -i –D /usr/local/pgsql/data  &
```

-i: Necesaria para que PostgreSQL escuche peticiones tcp en el puerto 5432, sin esta opción activada, el sistema de bases de datos sólo aceptaría conexiones locales, dentro del mismo host.

-D: Este parámetro le dice al postmaster que lo que sigue es el directorio que antes se creo (el database cluster) y donde está ese entorno donde PostgreSQL realizará las operaciones con la información. Esta es una manera simple de lanzar PostgreSQL.

pg_ctl

El comando pg_ctl es un wrapper[38], que simplemente llama a postmaster con unas opciones determinadas. Sus funcionalidades son parecidas al script que se ha creado para llamar a postmaster, da opción a lanzar/parar/relanzar el servidor.

Básicamente las opciones de pg_ctl son:

pg_ctl start

Se encarga de lanzar el daemon, ponerlo en background y redirección la salida de stdout y stderr a un archivo de log si se le pasa la opción -l.

pg_ctl stop

Se encarga de parar el daemon, dependiendo de la opción que se le pase.

pg_ctl restart

Esta opción llama a stop y luego a start sin más.

pg_ctl reload

Similar al restart, le envía una señal SIGHUP al server de forma que relea los ficheros de configuración.

[35] Un daemon es un programa que esta corriendo sin intervención humana, es decir, se están ejecutando constantemente en segundo plano esperando una instrucción para hacer alguna tarea en específico.
[36] Un fork se refiere a la creación de otro u otros procesos que son idénticos al que los creo, pero estos se llaman procesos hijos.
[37] Ponerse en background es pasar la aplicación que esta ejecutando en espera hasta que reciba las peticiones de los usuarios y así podamos utilizar la maquina para otros propósitos.
[38] Un Wrapper es un programa que llama a otro programa con opciones ya predeterminadas.

pg_ctl status

Comprueba si el postmaster esta corriendo y muestra tanto el PID, como los parámetros con los que se lanzó.

Usuarios y privilegios de la base de datos.

A la hora de utilizar el sistema gestor de bases de datos, lo mejor es crear un usuario que controlará tanto la creación de las bases de datos como de usuarios, que sea diferente al administrador que corre el PostgreSQL.

Lo primero que se debe hacer para la creación de un usuario es conectarse a la base de datos template1[39] la cual se crea una vez que se instala el manejador de base de datos, esta sirve como plantilla cuando se crean otras bases de datos. La conexión a la base de datos la hace de la siguiente forma:

```
# /usr/local/pgsql/bin/psql template1
```

Crear usuarios.

Para crear un usuario se utiliza la siguiente sentencia:

```
CREATE USER user_name
[WITH PASSWORD password]
[CREATEDB | NOCREATEDB]
[CREATEUSER | NOCREATEUSER]
[IN GROUP group1, ...groupN]
[VALID UNTIL 'abstime'];
```

Las opciones de este comando son las siguientes:

- *user_name:* Es el nombre del usuario. Este nombre sigue las reglas de que no se permiten caracteres especiales.
- *WITH PASSWORD:* Pide la contraseña del usuario. Si no se va a usar autentificación por password se puede omitir esta opción, de otra manera el usuario no será capaz de conectarse con el servidor de autenticación de passwords.
- *CREATEDB, NOCREATEDB:* Estas órdenes definen la capacidad de un usuario para que pueda o no crear bases de datos. Si se especifica CREATEDB, el usuario definido tendrá permiso para crear sus propias bases de datos. Usando NOCREATEDB se denegará a un usuario la capacidad de crear bases de datos. Si se omite esta opción, NOCREATEDB se usa por default.
- *CREATEUSER, NOCREATEUSER:* Esta opción determina si a un usuario se le permitirá crear nuevos usuarios. Esta opción hará del usuario un *superusuario* que podrá pasar por encima de todas las restricciones de acceso. Si se omite esta opción se creara NOCREATEUSER como valor por defecto del usuario.

[39] La base de datos template1 es la base que se crea al instalar el manejador de base de datos postgresql, además de servir como plantilla para las nuevas bases de datos que se crearan.

- *IN GOUP:* El nombre de un grupo dentro del cual se coloca al usuario como un nuevo miembro.

- *VALID UNTIL*: La orden VALIDO HASTA pone un valor absoluto a la fecha en la que la palabra clave del usuario pierde su validez. Si se omite esta orden el nombre del usuario valdrá para siempre.

PostgreSQL tiene otra forma de crear usuarios, el cual es un script *createuser* que tiene la misma funcionalidad que el comando CREATE USER (de hecho, llama a este comando) pero el script createuser puede ser ejecutado desde la línea de comandos, o sea, fuera del manejador de base de datos.

La sintaxis de este comando es:

```
createuser [ opciones ] [ nombre_usuario ]
```

Opciones:

-h, --host *host* : Especifica el nombre del host de la máquina sobre la que el postmaster corre.

-p, --puerto *puerto:* Especifica el puerto TCP/IP o el socket[40] local Unix sobre el que el postmaster atiende a las conexiones.

-e, --echo: Muestra las consultas que createuser genera y envía al backend.

-q, --quiet: No muestra respuesta alguna.

-d, --createdb: Permite al nuevo usuario crear bases de datos.

-D, --no-createdb: Impide al nuevo usuario crear bases de datos.

-a, --adduser: Permite al nuevo usuario crear otros usuarios.

-A, --no-adduser: Impide al nuevo usuario crear otros usuarios.

-P, --pwprompt: Si se especifica este parámetro, createuser mostrará un mensaje preguntando por el password del nuevo usuario. Esto no es necesario si no planea usar autentificación por password.

-i, --sysid *id_usuario*: Le permite elegir otro id de usuario que no sea el que se da por defecto. Esta opción es opcional.

Nombre_usuario: Especifica el nombre del usuario PostgreSQL que se va a crear. Este nombre debe ser único dentro de todos los existentes en PostgreSQL. Se le preguntará por un nombre y cualquier otra información que no se haya especificado en la línea de comandos.

Modificar usuarios.

Para poder cambiar el password y los privilegios de un usuario se utiliza el comando ALTER USER.

[40] Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión.

Se usa para cambiar los atributos de la cuenta de un usuario de PostgreSQL. Sólo un superusuario de una base de datos puede cambiar privilegios y fechas de caducidad de passwords con esta orden. Ordinariamente los usuarios sólo pueden cambiar su propia palabra clave.

```
ALTER USER user_name
[ WITH PASSWORD 'password' ]
[ CREATEDB | NOCREATEDB ] [ CREATEUSER | NOCREATEUSER ]
[ VALID UNTIL 'abstime' ]
```

Las opciones de este comando son las siguientes:

- *user_name:* El nombre del usuario cuyos atributos van a ser modificados.
- *WITH PASSWORD:* El nuevo password que va a ser usado en la cuenta.
- *CREATEDB, NOCREATEDB:* Estas cláusulas definen la capacidad de un usuario para crear o no bases de datos. Si se especifica CREATEDB, el usuario podrá definir sus propias bases de datos. Usando NOCREATEDB se deniega a un usuario la capacidad de crear bases de datos.
- *CREATEUSER, NOCREATEUSER:* Estas cláusulas determinan si un usuario está autorizado a crear nuevos usuarios él mismo. Está opción hace ser además al usuario un superusuario que puede pasar por encima de todas las restricciones de acceso.
- *VALID UNTIL:* La fecha en la que el password del usuario expirará.

Borrar usuarios.

Este comando borra de la base de datos el usuario creado. Cabe aclarar que esta cláusula no borra tablas, vistas u otros objetos que pertenezcan al usuario. Si el usuario es dueño de una base de datos, se producirá un error.

Para borrar a un usuario existente, se utiliza el comando DROP USER y se especifica el nombre del usuario que se quiera borrar:

```
DROP USER user_name;
```

Al igual que para la creación de usuarios Postgresql cuenta con un script para el borrado de usuarios desde fuera del manejador de base de datos el cual es *dropuser* tiene la misma función de DROP USER (de hecho, invoca este comando).

```
dropuser [ nombre_usuario ]
```

Nombre_usuario: Especifica el nombre de usuario PostgreSQL que va a ser borrado. Este nombre debe existir en la instalación Postgresql. Se le preguntará un nombre si no se ha especificado ninguno en la línea de comandos.

Grupos.

Como en Unix, los grupos son una manera lógica de agrupar a usuarios para facilitar la administración de privilegios: los privilegios se pueden conceder (GRANT), o revocar (REVOKE), de un grupo en general. Para crear a un grupo se utiliza:

```
CREATE GROUP name
[ WITH
[ SYSID gid ]
[ USER  username [, ...] ] ]
```

Las opciones de este comando son:

- *name*: El nombre del grupo.
- *gid*: La cláusula SYSID puede ser usada para elegir el número id del grupo PostgreSQL del grupo nuevo. El uso de esta cláusula es opcional. En caso de no especificar el número id del grupo, se asignara el número mayor ya asignado, empezando por 1.
- *username*: Una lista de los usuarios a incluir en el grupo. Los usuarios tienen que existir antes de incluirlos en el grupo.

Para agregar a usuarios o quitar usuarios de un grupo, se utiliza respectivamente:

```
ALTER GROUP name ADD USER nombre de usuario1, ... ;
ALTER GROUP name DROP USER nombre de usuario1, ... ;
```

Sólo los administradores de bases de datos pueden usar este comando. Añadir un usuario a un grupo no crea ese usuario. Igualmente, eliminar a un usuario de un grupo no significa que se elimine al usuario en si mismo.

- *nombre*: El nombre del grupo a modificar.
- *nombre de usuario*: Usuarios que van a ser añadidos o eliminados del grupo. Los nombres de usuarios deben existir.

Privilegios.

Cuando se crea un objeto en la base de datos, se le asigna un dueño. El dueño es el usuario que ejecutó la declaración de la creación. Para cambiar al dueño de una tabla, índice, secuencia, vista, se utiliza el comando *ALTER TABLE*. Por default, sólo el dueño (o superusuario) puede hacer cualquier cosa con el objeto. Para permitir que otros usuarios lo utilicen, los privilegios deben ser concedidos (GRANT).

Otorgar privilegios.

Hay varios privilegios: SELECT, INSERT, UPDATE, DELETE, RULE, REFERENCES, TRIGGER, CREATE, TEMPORARY, EXECUTE, USAGE, y ALL PRIVILEGES. El hecho de modificar o destruir un objeto es siempre el privilegio del dueño solamente, para ello se utiliza el comando GRANT.

La sintaxis de este comando es la siguiente:

```
GRANT privilege [, ...] ON object [, ...]
TO { PUBLIC | GROUP group | username }
```

Las opciones posibles para los privilegios son:

- *SELECT:* Acceso a todas las columnas de una tabla/vista específica.
- *INSERT:* Inserta datos en todas las columnas de una tabla específica.
- *UPDATE:* Actualiza todas las columnas de una tabla específica.
- *DELETE:* Elimina filas de una tabla específica.
- *RULE:* Define las reglas de la tabla (CREATE RULE).
- *ALL:* Otorga todos los privilegios.

El nombre de un objeto al que se quiere conceder el acceso. Los posibles objetos son:

- Tabla
- Vista
- Secuencia
- Índice

Y las opciones para a quien se le va a otorgar los privilegios son:

- *PUBLIC:* Una abreviación para representar a todos los usuarios.
- *GROUP group:* Un *grupo* al que se otorgan privilegios. En la actual versión, el grupo debe haber sido creado.
- *username:* El nombre de un usuario al que se quiere conceder privilegios. PUBLIC es una abreviatura para representar a todos los usuarios.

Una vez que un usuario tiene privilegios sobre un objeto, tiene posibilidad de ejecutar ese privilegio. No hay necesidad de conceder privilegios al creador de un objeto; el creador obtiene automáticamente TODOS los privilegios, y puede también eliminar el objeto.

Se usa \z para obtener más información sobre los permisos de los objetos existentes.

Análogamente existe un comando para quitar privilegios a los usuarios sobre los diferentes objetos, este comando es REVOKE.

Revocar privilegios.

La sintaxis de este comando es la siguiente:

```
REVOKE privilegio [, ...]
ON objeto [, ...]
FROM { PUBLIC | GROUP | nombre_usuario }
```

Los posibles privilegios que pueden ser revocados son:

SELECT: Privilegio para acceder a todas las columnas de una tabla o vista específica.
INSERT: Privilegio de insertar datos en todas las columnas de una tabla específica.
UPDATE: Privilegio para actualizar todas las columnas de tabla.
DELETE: Privilegio para borrar filas de una tabla específica.
RULE: Privilegio para definir reglas en una tabla o vista.
ALL: Otorga todos los privilegios.

El nombre de un objeto sobre el que revocar el acceso. Los posibles objetos en los cuales se pueden revocar privilegios son:

- Tabla
- Vista
- Secuencia
- Índice

Y se le pueden revocar los privilegios, ya sea a:

- *PUBLIC*: Para especificar todos los usuarios.
- *GRUPO:* El nombre de un grupo al cual se le revoca privilegios.
- *Nombre_usuario*: El nombre de un usuario al cual se revoca privilegios.

Manejo de bases de datos.

Para el manejo de base de datos primero hay que crearlas para poder utilizarlas, y ahí poder crear los objetos.

Crear base de datos.

Para crear una base de datos se utiliza el siguiente comando, el cual creará una nueva base de datos, y el creador de está pasa a ser el propietario de la nueva base de datos.

```
CREATE DATABASE name
[[WITH OWNER = dbowner]
[LOCATION = dbpath]
[TEMPLATE = template]
[ENCODING = encoding]]
```

Las opciones que puede tener son:

- *name*: Es el nombre de la base de datos a crear.
- *WITH OWNER*: Especifica el usuario que será el dueño de la base de datos, esta opción sólo puede ser utilizada por un superusuario, ya que es el único que puede cambiar el dueño de la base de datos, si se omite el dueño es como ya se había dicho el usuario que crea la base.
- *LOCATION:* Es una ubicación alternativa para almacenar la nueva base de datos en el sistema de archivos.
- *TEMPLATE:* Se especifica si se quiere utilizar alguna base de datos como platilla para la creación de la nueva base de datos.
- *ENCODING:* Aquí se especifica el tipo de set de caracteres que se ha de utilizar para la base de datos.

Se ha de tener el privilegio especial CREATEDB para crear bases de datos.

Existe otra forma de crear una base de datos, esto se hace con el comando *createdb* el cual es un script shell[41] construido y basado en el comando CREATE DATABASE.

[41] El shell es el intérprete de comandos de Linux, además de ser la interfaz entre el sistema operativo y el usuario.

Su sintaxis es la siguiente:

```
createdb [ opciones ] dbname
```

Opciones:

- -D, --location *datadir*: Especifica la localización alternativa de la base de datos. Esta es la localización de las tablas del sistema, no la localización de esta base de datos en específica, que puede ser diferente.
- -O – Owner: Se especifica el dueño de la base de datos.
- -E, --encoding *encoding*: Especifica el esquema de codificación de caracteres que se usara con esta base de datos.

dbname: Especifica el nombre de la base de datos que será creada. El nombre debe ser único entre todas las bases de datos PostgreSQL. El valor por omisión es crear una base de datos con el mismo nombre que el usuario en curso del sistema.

Así como existe un comando para la creación de bases de datos, también existe uno para poder borrar las bases de datos.

Borrar base de datos.

La sintaxis que utiliza este comando es la siguiente:

```
DROP DATABASE name
```

name: El nombre de la base de datos existente que se desea eliminar.

Al igual que para crear una base de datos se debe tener el privilegio especial CREATEDB para eliminar bases de datos. Esta orden no puede ser ejecutada mientras se está conectado a la base de datos. Por lo tanto, puede ser más conveniente usar el shell script *dropdb*, que emplea este comando.

Su sintaxis es la siguiente:

```
dropdb nombre_bd
```

nombre_bd: Especifica el nombre de la bases de datos que va a ser borrada. Debe ser una de las existentes dentro de Postgresql.

Plantillas de bases de datos.

El comando CREATE DATABASE realmente trabaja copiando una base de datos existente. Por default, copia la base de datos estándar del sistema llamada template1. Así esta base de datos es el "template" para crear nuevas bases de datos. Si se agregan objetos a la base de datos template1, estos objetos serán copiados en bases de datos creadas posteriormente por el usuario. Este comportamiento permite modificaciones al sistema estándar de objetos en bases de datos.

Por ejemplo, si se instala el lenguaje procedural PL/pgSQL[42] en template1, estará automáticamente disponible en las bases de datos del usuario sin ninguna acción adicional que es tomada cuando se hacen esas bases de datos.

Hay una segunda base de datos estándar del sistema llamada template0. Esta base de datos contiene los mismos datos que el contenido inicial de template1, esto es, solamente los objetos estándares predefinidos por la versión de PostgreSQL. template0 sin que nunca se cambie después initdb.

Para crear una base de datos copiando template0 debemos usar:

```
CREATE DATABASE dbname TEMPLATE template0;
```

ó `createdb -T template0 dbname`

Conjunto de caracteres.

PostgreSQL brinda el soporte de caracteres desde dos puntos de vista:

- Usando las características locales del sistema operativo para proporcionar especificaciones locales sobre orden de colocación, formato de números, traducción de mensajes y otros aspectos.

- Proporcionando un número de juegos de caracteres diferentes definidos en el servidor de PostgreSQL, incluyendo juegos de caracteres multiple-byte, para soportar el almacenamiento de texto de todo tipo de idiomas y proporcionando juego de caracteres para la traducción entre el cliente y el servidor.

Soporte local.

El soporte local de caracteres se refiere al uso de preferencias culturales con respecto a los alfabetos, ordenamientos, formato de números, etc. PostgreSQL utiliza los estándares ISO C y POSIX[43] proporcionadas por el sistema operativo del servidor.

El soporte local de caracteres se inicializa automáticamente cuando se crea un cluster de la base de datos usando initdb. Initdb inicializará el cluster de la base de datos con las opciones locales de su ambiente de ejecución; así si el sistema esta listo para utilizar el soporte local de caracteres que se desea en el cluster de la base de datos no hay nada más que hacer.

Si se desea utilizar un soporte local de caracteres diferente, se puede indicar a initdb exactamente que soporte se desea con la opción -- locale. Por ejemplo:

```
initdb -- locale=sv_SE
```

Este ejemplo fija el soporte de caracteres local al sueco (sv). Otras posibilidades pueden ser en_US (ingles de ESTADOS UNIDOS) y fr_CA (francés de Canadá).

---

[42] El lenguaje PL/pgSQL es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL. Permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones.

[43] POSIX (Portable Operating System Interface para UNIX). Familia de estándares relacionados especificados por la IEEE para definir APIs para la compatibilidad de software entre los diferentes sistemas operativos Unix.

Los datos por default que son elegidos por el initdb se escriben en el archivo postgresql.conf para funcionar como valores por default cuando el servidor es iniciado.

Si se suprimen las asignaciones de postgresql.conf entonces el servidor heredará los valores del ambiente de ejecución.

Una de las ventajas que proporciona el soporte local de caracteres es que ayuda particularmente en el ordenamiento de los querys usando ORDER BY.

Soporte de juego de caracteres.

El soporte de juego de caracteres en PostgreSQL permite almacenar texto en diferentes juegos de caracteres, incluyendo juegos de caracteres single-byte tales como la ISO 8859 y del multiple-byte tales como EUC (Extended Unix Code), Unicode. Todos los juegos de caracteres se pueden utilizar transparentemente a través del servidor. El juego de caracteres por default es seleccionado mientras se inicializa el cluster de la base de datos PostgreSQL. Este puede ser eliminado cuando se crea una base de datos usando el comando createdb o usando el comando sql CREATE DATABASE. Se pueden tener múltiples bases de datos cada una con un juego de caracteres diferentes.

Tareas de mantenimiento de la base de datos.

Existen algunas tareas de mantenimiento que se deben realizar periódicamente para asegurar que el servidor PostgreSQL se ejecute sin problemas.

Una tarea obvia del mantenimiento es la creación de respaldos de los datos bajo un determinado horario. Sin un respaldo reciente, no existe ninguna posibilidad de recuperar los datos después de una catástrofe, por ejemplo fuego, falla del disco, entre otros.

Una segunda categoría de las tareas de mantenimiento periódicas es la conocida como “VACUUMING”.

LIMPIAR CON VACUUM

El comando VACUUM de PostgreSQL se debe correr regularmente por varias razones:

1. Para recuperar espacio en disco ocupado por actualizaciones o registros borrados.
2. Para actualizar la estadística de datos usada por el planificador de consultas en PostgreSQL.

La frecuencia y el alcance de las operaciones de VACUUM realizadas por cada una de estas razones variarán dependiendo de las necesidades de cada sitio. Por lo tanto, los administradores de la base de datos deben entenderlas para desarrollar una estrategia de mantenimiento adecuada.

Comenzando en PostgreSQL 7.2, la forma estándar de VACUUM puede funcionar en paralelo a operaciones normales de la base de datos, por ejemplo, selects, inserts, updates, pero no cambia la definición de la tabla. La rutina de VACUUMING no esta tan implícita como en versiones anteriores.

En una operación normal de PostgreSQL, un UPDATE o DELETE de registro no lo quita inmediatamente.

Este enfoque es necesario para obtener los beneficios del control de concurrencia multiversion: la versión de la fila no debe ser suprimida mientras siga siendo visible para otras transacciones. Pero eventualmente, una versión vieja se vuelve obsoleta para cualquier transacción. El espacio que ocupa se debe reutilizar para los nuevos registros, esta operación esta a cargo de VACUUM.

Claramente, una tabla que recibe updates o deletes constantemente necesitará ser limpiadas con VACUUM más a menudo que las tablas que son raramente actualizadas. Puede ser útil habilitar tareas periódicas con el demonio de cron para que limpien con VACUUM las tablas que sufren updates o deletes frecuentes. Esto puede ser útil si dentro del sistema existen tablas grandes ya que el aplicar VACUUM a pequeñas tablas no es muy recomendable.

El objetivo estándar de VACUUM es mantener en uso un pequeño espacio de disco. La función estándar es que al encontrar versiones viejas de los registros hace disponible el espacio utilizado por el mismo para nuevos registros, pero no intenta modificar el archivo de tabla o regresar el espacio de disco al sistema operativo. Si se necesitará regresar espacio de disco al sistema operativo se tendría que ejecutar un VACUUM completo, pero cuál es el caso de regresar espacio que será nuevamente asignado. Generalmente es mejor ejecutar un VACUUM estándar ya que el VACUUM completo se utiliza para tablas frecuentemente actualizadas.

Si se tiene una tabla cuyo contenido ha sido borrado totalmente, considere hacerlo mediante el comando TRUNCATE en lugar de DELETE seguido por VACUUM.

Actualización del planificador estadístico.

El planificador de consultas de PostgreSQL hace uso de la información estadística sobre el contenido de tablas para optimizar las consultas. Estas estadísticas son recopiladas por el comando ANALYZE, que se puede invocar así mismo o como un paso opcional en VACUUM. Es importante contar con estadísticas exactas, ya que de lo contrario el rendimiento de las consultas puede no ser el mejor.

Al igual que con VACUUM para recuperar espacio, las actualizaciones frecuentes de las estadísticas son más útiles para las tablas constantemente actualizadas que para las poco actualizadas. Pero para una tabla muy grande, no existe necesidad de actualizaciones de la estadística si la distribución estadística de los datos no está cambiando mucho.

Es posible correr ANALYZE sobre tablas específicas e incluso columnas de una tabla, así que existe una gran flexibilidad para poner al día alguna estadística en específico si así se requiere. En la práctica, sin embargo, la utilidad de esta característica es muy poco usada.

La práctica recomendada para la mayoría de los sitios es programar un ANALYZE a toda la base de datos una vez al día en una hora de poco trabajo; esto se puede combinar con VACUUM cada noche.

Respaldos y restauración de información.

En Postgres las bases de datos se deben respaldar regularmente, ya que la información como ya se sabe es lo mas importante para las empresas. El procedimiento es muy simple, es importante tener un bosquejo básico de las técnicas de respaldo.

Hay dos funciones fundamentales para trabajar con PostgreSQL:

- Respaldo de SQL
- Respaldo a nivel del sistema de archivos

Respaldo a nivel de sistema de archivos.

Una alternativa para respaldar, es copiar directamente los archivos donde PostgreSQL almacena los datos de la base de datos. Se puede utilizar cualquier método para los respaldos generalmente del sistema de archivos, se hace de la siguiente manera:

```
tar -cf backup.tar /usr/local/pgsql/data
```

Hay dos restricciones, que hacen este método impráctico, o por lo menos inferiores al método pg_dump :

El servidor de la base de datos debe cerrarse para conseguir un respaldo usable. Para servidores que tienen mucha demanda seria inusual este método por el número de peticiones. Cabe mencionar que para hacer la restauración de datos es necesario cerrar el demonio.

La disposición del sistema de archivos no es tan dinámica. Esto quiere decir que no se puede hacer la restauración de archivos (información) de sólo unas tablas (archivos) o bases de datos individuales (directorios) por que la mitad de la información esta en los archivos o directorios respectivamente y la otra mitad esta en el pg_clog.

Aunque sólo se restaure unos archivos (tablas), el resto de las tablas tendrían problemas por que ya no coincide con el pg_clog.

Respaldo de SQL.

La idea básica del método de SQL-dump es generar un archivo del texto con sentencias SQL, el objetivo es reconstruir la base de datos en el mismo estado que era a la hora de la respaldo.

PostgreSQL nos proporciona las herramientas pg_dump y pg_dumpall, las cuales ayudan tanto como para respaldar las tablas, como para copiarlas de un sistema a otro en un formato transportable. En algunos casos al actualizar la versión de PostgreSQL será necesario primero respaldarlas con estas herramientas y para posteriormente volverlas a cargar.

pg_dump: Se emplea para respaldar una base de datos o una tabla en particular.
pg_dumpall: Respalda todas las bases de datos en el sistema.

El uso básico de este comando es:

```
pg_dump dbname > outfile
```

El comando pg_dump es utilizado por un cliente regular de postgres. Esto significa que se puede hacer el procedimiento de respaldo de cualquier host remoto que tenga acceso a la base de datos. Pero se debe recordar que pg_dump no funciona con permisos especiales. Se debe tener permisos de lectura en todas las tablas que va a respaldar, la mayoría de las veces tiene que ser el dueño de la base de datos.

Respaldos creados por el pg_dump son internamente constantes, es decir, si corremos un respaldo y la base de datos sigue actualizándose el respaldo comprende hasta el momento que efectuó el respaldo, cabe mencionar que no bloquea tablas.

Restauración de la descarga.

Los archivos del texto creados por el pg_dump deben ser leídos adentro por programa psql. La forma general del comando para restaurar un respaldo es:

```
psql dbname < infile
```

Donde infile es lo que se utiliza como outfile con el comando pg_dump. La base de datos dbname NO será creado por este comando, hay que crearla antes de ejecutar el psql.

Una vez que esté restaurado, es necesario ejecutar ANALYZE en cada base de datos, el optimizador saca estadísticas. O correr vacuumdb - a - z para analizar todas las bases de datos.

pg_dump y psql tienen capacidad para leer y escribir en pipe (|) esto permite respaldar una base de datos directamente a partir de un servidor a otro; por ejemplo:

```
pg_dump - h host1 dbname |psql - h host2 dbname
```

Usar el pg_dumpall..

Este mecanismo genera un respaldo de todas las bases de datos usuarios y grupos. Para generar un respaldo de todo esto basta con ejecutar el siguiente comando.

```
pg_dumpall > outfile
```

El respaldo se puede restaurar con el psql:

```
psql template1 < infile
```

1.6.5. Administración de Bases de Datos en MySQL

Instalación y configuración de MySQL.

Lo primero que se debe hacer es bajar el paquete de mysql. Una vez que se cuenta con el paquete se ejecutan las siguientes líneas:

```
# groupadd mysql
# useradd -g mysql mysql
# tar -zxvf /ruta/hacia/mysql-4.1.12.tar.gz
# cd mysql-4.1.12
# ./configure --prefix=/usr/local/mysql
# make
# make install
# cp /usr/src/mysql-4.1.12/support-files/mysql.server /etc/rc.d/init.d/mysqld
# chmod 755 /etc/rc.d/init.d/mysqld
# ln -s /etc/rc.d/init.d/mysqld /etc/rc.d/rc5.d/S99mysql
# ln -s /etc/rc.d/init.d/mysqld /etc/rc.d/rc0.d/K01mysql
```

```
# cp /usr/local/mysql/share/mysql/my-small.cnf /etc/my.cnf
# cd /usr/local/mysql
# bin/mysql_install_db --user=mysql
# chown -R root  .
# chown -R mysql var
# chgrp -R mysql .
# /etc/init.d/mysqld start
# /usr/local/mysql/bin/mysqladmin -u root password <password>
# /usr/local/mysql/bin/mysql -u root -p
```

Manejo del proceso del servidor MySQL.

Para utilizar una base de datos, el daemon de mysqld debe estar ejecutándose; para iniciarlo es necesario ser root[44] y teclear alguna de las siguientes líneas:

```
# /etc/init.d/mysqld start
# service mysqld start
# /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

En la ultima línea, el comando mysqld_safe arranca el servidor; el símbolo "&" indica que el programa se ejecutara en segundo plano. Para asegurar de que el demonio de mysql se está ejecutando, se puede teclear alguna de las siguientes líneas:

```
# ps –fea | grep mysql | grep ^root | grep –v grep
# /usr/local/mysql/bin/mysqladmin –p ping
```

Para dar de baja el demonio de mysql se teclea una de las siguientes líneas:

```
# /etc/init.d/mysqld stop
# service mysqld stop
# /usr/local/mysql/bin/mysqladmin --user=root -p shutdown
```

Esto detendrá de forma segura el motor. Otra manera más drástica de detenerlo es mediante el uso del comando kill, no es muy recomendable ya que puede corromper los datos.

Usuarios y privilegios de la base de datos.

Agregar usuarios.

Para permitir a otros usuarios utilizar una base de datos desde su máquina local, el usuario debe estar dado de alta en muchos lugares. El RDBMS de MYSQL contiene una base de datos llamada mysql. Ésta contiene los permisos para todas las bases de datos de mysql y ésta formada por las siguientes tablas:

[44] Root es el súper usuario del sistema operativo Linux, este usuario se utiliza para la administración del sistema.

- *user*: Contiene los nombres, contraseñas, hosts y privilegios de los usuarios de MySQL.
- db: Contiene todos los usuarios, bases de datos y nombres de hosts de MySQL.
- host: Contiene todos los nombres de hosts, bases de datos y privilegios de MySQL.

Para que alguien pueda usar la base de datos, el nombre de la máquina desde donde se está conectando debe existir en la tabla host. El usuario debe existir en la tabla user y la base de datos debe existir en la tabla db.

Para agregar el nombre del host y la base de datos en la tabla host debe usar el monitor de MYSQL:

```
# /usr/local/mysql/bin/mysql –p
```

Debe hacer que la base de datos mysql sea la activa:

```
mysql> USE mysql;
```

Para añadir simultáneamente el nombre del host y la base de datos al RDBMS mysql, debe usar el comando *INSERT*:

```
mysql> INSERT INTO host VALUES(‘localhost’, ‘bd_muestra’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ’y’, ‘y’);
```

Para añadir un usuario a la tabla user:

```
mysql> INSERT INTO user VALUES('localhost', 'usuario_prueba', PASSWORD('pas123'), 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', '', '', '', '', 0, 0, 0);
```

La función PASSWORD es una función intrínseca, es decir, puede llamarse desde MySQL. Esta función recibe una cadena como argumento y la encripta.

Para añadir la base de datos y los usuarios a la base de datos mysql:

```
mysql> INSERT INTO db VALUES('localhost', 'bd_muestra', 'usuario_prueba', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y', 'y','y');
```

Por ultimo es necesario correr el siguiente comando para que la tabla de privilegios se actualice y los cambios surtan efecto.

```
mysql> FLUSH PRIVILEGES;
```

Otra forma de crear cuentas de usuario es mediante el siguiente comando:

```
mysql> GRANT ALL PRIVILEGES ON bd_prueba.* TO ‘usuario_prueba’@’localhost’ IDENTIFIED BY ‘pas123’ WITH GRANT OPTION;
```

Seguridad.

El sistema de seguridad de MySQL puede asignar diferentes niveles de acceso a los usuarios que van desde la posibilidad de registrarse desde una máquina específica como un usuario específico hasta acceso completo de administrador desde cualquier lugar.

MySQL mantiene todos los permisos y privilegios en la base de datos mysql. Ésta es una de las dos bases de datos que se crean automáticamente cuando se instala MySQL (la otra es test). Las únicas personas que deberían tener acceso a ella son los administradores de la base de datos, mysql es igual que cualquier otra base de datos de MySQL. Las tablas principales de la base mysql son:

- user
- db
- host

Estas tablas se conocen como tablas de permisos de acceso. Cada columna de estas tablas muestra los permisos que tiene una persona ya sea con una "Y" (que significa que pueden realizar la operación) o una "N" (que significa lo contrario).

La tabla user

La tabla user contiene los datos de los permisos de todos los usuarios que tienen acceso a mysql.

La tabla db

Contiene los permisos para todas las bases de datos que contiene el servidor mysql. Los permisos otorgados aquí son proporcionados solamente para la base de datos mencionada.

La tabla db tiene prácticamente las mismas columnas que la tabla user, con algunas excepciones. Debido a que esta tabla rige los permisos en el nivel de la base de datos, no hay privilegios del nivel de administrador, tales como reload_priv, shutdown_priv, process_priv y file_priv. Estos permisos no se relacionan con las operaciones que pueden realizarse en las bases de datos, así que dichos permisos sólo se encuentran en la tabla user.

La tabla host

Esta tabla, junto con la tabla db, controla el acceso al limitar los hosts que pueden conectarse a la base de datos. Esta tabla contiene las mismas columnas que la tabla db.

Etapas de control.

Existe una jerarquía de seguridad en el sistema de bases de datos de mysql.

- Cuando un usuario se conecta a la base de datos:

    1. Mysql primero busca en la tabla user para verificar si puede encontrar una coincidencia para el nombre de host, nombre de usuario y contraseña.
    2. Si la encuentra, se le proporciona al usuario acceso al sistema.

- Cuando éste envía una consulta a la base de datos, mysql:

    1. Primero consulta la tabla user para ver qué privilegios tiene.
    2. Si el usuario no tiene privilegios en dicha tabla, consulta la tabla db. Nuevamente, verifica una coincidencia en la tabla con el nombre de host, nombre de usuario y base de datos. Si la encuentra, consultara los privilegios que tiene la persona.

    3. Si no tiene los privilegios necesarios para la consulta, mysql explorará la tabla tables_priv y después la tabla columns_priv en busca de los permisos necesarios para ejecutar la consulta.
    4. Si no puede encontrar algún permiso, generará un error. Todo esto sucede cada vez que se realiza una consulta en mysql.

Como se puede observar, hay dos puntos de control. Uno es la verificación de la conexión y el segundo punto de control es la verificación de la solicitud.

Privilegios.

Si no le agrada actualizar directamente las tablas de permisos de acceso, existe otra forma para administrar los privilegios de los usuarios: mediante el comando GRANT.

Otorgar privilegios.

La sintaxis de este comando es la siguiente:

```
GRANT [privilegios]
ON nombrebasededatos.nombredetabla
TO nombreusuario@nombredehost
IDENTIFIED BY "contraseña"
[WITH GRANT options]
```

Después de la palabra clave GRANT, debe listar todos los privilegios que desea conceder al nuevo usuario. Los privilegios que puede conceder son:

- *ALL:* Proporciona al usuario todos los privilegios disponibles.
- *ALTER:* Permite que el usuario modifique tablas, columnas e índices.
- *CREATE:* Permite que el usuario cree tablas o bases de datos.
- *DELETE:* Permite que el usuario borre registros de las tablas.
- *DROP:* Permite que el usuario borre tablas o bases de datos
- *FILE:* Permite que el usuario lea y escriba archivos en el servidor.
- *INDEX:* Permite que el usuario agregue o elimine índices.
- *INSERT:* Permite que el usuario agregue registros a la base de datos.
- *PROCESS:* Permite que el usuario visualice y cancele procesos del sistema de mysql.
- *REFERENCES:* Sin uso actualmente en mysql
- *RELOAD:* Permite que el usuario utilice la instrucción flush.
- *SELECT:* Permite que el usuario efectúe consultas SELECT.
- *SHUTDOWN:* Permite que el usuario apague el servidor mysql.

- *UPDATE:* Permite que el usuario edite registros existentes en la base de datos.
- *USAGE:* Permite que el usuario se conecte al servidor. Este tipo de usuario no cuenta con privilegio alguno.

Después de la palabra clave ON, debe listar la base de datos y la tabla o tablas a las cuales desea aplicar estos privilegios. Para especificar todas las tablas en la base de datos, puede utilizar un asterisco: bd_muestra.*

Si quisiera aplicar estos permisos globalmente, tendría que utilizar dos asteriscos: *.*

La siguiente parte del comando es la cláusula TO. Aquí se establece el usuario y su nombre de host.

La siguiente parte de la instrucción es la contraseña. No necesita utilizar la función password en la instrucción GRANT. MySQL lo hace automáticamente. La última parte del comando es completamente opcional. Le ofrece al usuario especificado la capacidad de utilizar también la instrucción GRANT.

Eliminación de usuarios y revocación de privilegios.

Hay dos formas para eliminar a un usuario, la primera es la edición manual de las tablas de permisos de acceso. Se envía una instrucción DELETE para cada una de las tablas en la base de datos de mysql. Para hacerlo de una manera correcta y segura, debe especificar el usuario y nombre de host en la cláusula WHERE. Por ejemplo, utilizaría los siguientes comandos para eliminar a juan de las tablas de permisos de acceso:

```
DELETE FROM user WHERE user=”juan” and host=”%”;
DELETE FROM bd WHERE user=”juan” and host=”%”;
```

La segunda es por medio de la instrucción revoque:

```
REVOQUE [privilegios] ON BASEDEDATOS.tabla FROM usuario@host
```

Los privilegios son los mismos que se utilizan en la cláusula GRANT.

Manejo de bases de datos.

Creación de una base de datos.

La forma de crear una base de datos es utilizando el comando *CREATE*, este comando permite crear diversos objetos.

```
mysql> CREATE DATABASE <nombre_base_datos>;
```

Una vez creada la base de datos se puede empezar a usarla, para eso es necesario utilizar el comando, USE, por ejemplo:

```
mysql> USE <nombre_base_datos>;
```

Eliminar una base de datos.

La forma de eliminar una base de datos es utilizando el comando *DROP*, este comando permite borrar diversos objetos.

```
mysql> DROP DATABASE <nombre_base_datos>;
```

Administración de una base de datos con mysqladmin.

La utilería mysqladmin[45] se utiliza para realizar una gran variedad de operaciones administrativas en una base de datos de MySQL. Normalmente se ejecuta desde la línea de comandos.

Al invocarla, debe transferir algunas opciones y comandos a mysqladmin que le indiquen como ejecutarse y que hacer. La sintaxis para invocar al comando mysqladmin es:

```
mysqladmin [opciones] comando1 [opc_com1] comando2 [opc_com2]
```

Tareas de mantenimiento de la base de datos.

Existen diversas tareas de mantenimiento para tratar que nuestro sistema manejador de base de datos se encuentre en óptimas condiciones.

Afinar el rendimiento.

Cuándo se tienen problemas de rendimiento, uno de los primeros lugares en donde se debe buscar es en el sistema mismo. ¿En qué tipo de computadora se está ejecutando su servidor de base de datos?, ¿Cuánta memoria tiene?, ¿Cuál es la velocidad del procesador?

MySQL se desarrollo en máquinas Intel que ejecutan Linux, motivo por el que Linux es la mejor plataforma en la que puede ejecutarse.

MySQL es capaz de ejecutar varios subprocesos a la vez. Esto significa que cada vez que se realiza una conexión. MySQL crea un subproceso. Cada subproceso consume memoria. El almacenamiento en caché[46] de los resultados también consume memoria. Por lo tanto, entre más memoria mejor. Esto ayuda a mejorar el rendimiento. También hay que tomar en consideración el disco duro. Un disco duro rápido genera resultados más rápidos.

Establecimiento de las variables del sistema

Después de tener el mejor hardware, lo siguiente es optimizar el sistema de la base de datos. Existen muchas variables que controlan la forma de operación de MySQL. Para ver la configuración actual:

1. Asegúrese de estar en el directorio mysql y que el servidor esta ejecutándose y teclee:

```
bin/mysqladmin –p variables
```

[45] Mysqladmin es una herramienta para la administración del manejador de base de datos MYSQL.
[46] La memoria caché es una clase de memoria RAM estática (SRAM) de acceso aleatorio y alta velocidad, situada entre el CPU y la RAM; se presenta de forma temporal y automática para el usuario, que proporciona acceso rápido a los datos de uso más frecuente.

Pedirá la contraseña de root y mostrara las variables del sistema de MySQL. Para cambiar cualquier valor, utilice la siguiente sintaxis:

```
safe_mysql –O variable = valor
```

Variables más importantes para optimizar el sistema:

- *back_log:* Controla cuantas conexiones puede tener MySQL mientras esta realizando nuevos subprocesos.
- *delayed_queue_size*: Controla el número de filas que pueden ser almacenadas en cola de espera mientras se utiliza un comando INSERT DELAYED. Se debe de aumentar si se ponen muchas inserciones en la cola de espera.
- *flush_time:* Controla la cantidad de tiempo en segundos, antes de que MySQL escriba en el disco lo que esta almacenando en caché. Entre más tenga operaciones de E/S al disco, más lenta se volverá la base de datos. Aumente este valor para retrasar la escritura en el disco del contenido del caché.
- *table_cache:* Controla el número de tablas abiertas para todos los subprocesos. Si incrementa aumentara el número de tablas que puede abrirse al mismo tiempo. Con esto disminuye la cantidad de sobre carga y podría acelera el proceso.
- *waite_timeout:* Controla la cantidad de tiempo antes de que una conexión se cierre por inactividad. Un número menor puede incrementar la velocidad.
- *buffers:* Al incrementar cualquiera de los búferes se acelerará su base de datos. Si aumenta mucho los búferes puede tener efecto contrario. Esta configuración se fundamenta en la cantidad de memoria disponible.

Mejorar el rendimiento con el buen uso de instrucciones SQL.

Una mala estructura de la base de datos puede ocasionar problemas, se debe crear un buen diseño de base de datos y de las instrucciones SQL que utiliza para acceder a los datos. Algunas reglas para lograr tablas más rápidas son:

- Utilizar el tipo de datos más pequeño posible.
- Obligue a que se pongan valores en las columnas utilizando NOT NULL.
- Evite el uso de columnas de longitud variable.
- Evite utilizar demasiados índices.
- Elija el tipo de tabla tomando en cuenta el rendimiento.
- Utilice los valores predeterminados en las columnas.

Reglas para crear mejores consultas SQL:

- Escribir las consultas de manera que puedan utilizar índices.
- Utilizar la palabra clave LIMIT en las consultas.
- No utilizar puntuación extraña siempre que sea posible, porque MySQL tendría que analizar los signos de puntuación extraños antes de realizar la consulta.
- La seguridad es directamente proporcional a la cantidad de sobrecarga creada al ejecutar consultas. A mayor seguridad se crea más sobrecarga. No sacrifique la seguridad por el rendimiento.
- Si una tabla tiene un índice se debe utilizar. Las condiciones numéricas se ejecutan más rápido que las de cadenas.

Respaldos y restauración de información.

Mysqldump.

La utilería mysqldump comparte algunas opciones comunes de mysqlimport, pero esta utilería hace algunas cosas más. Toma toda la base de datos y la vacía en un solo archivo de texto. Este archivo contiene todos los comandos de SQL necesarios para recrear su base de datos. Toma el esquema y lo convierte a la sintaxis de DDL adecuada (instrucciones CREATE), y toma todos los datos y crea instrucciones INSERT. Esta utilería hace ingeniería inversa de su base de datos. Debido a que todo está contenido en el archivo de texto, puede ser importado de nuevo en su base de datos de MySQL con un simple procesamiento por lotes y la sintaxis adecuada de SQL. Para vaciar su base de datos:

```
bin/mysqldump -p bd_muestra > respaldo.txt
```

Esta utilería también le permite especificar la tabla en la que quiere hacer el vaciado, si sólo quisiera vaciar el contenido de cierta tabla de la base de datos bd_muestra en un archivo tendría que especificarlo de la siguiente manera:

```
bin/mysqldump -p bd_muestra nombre_tabla > respaldo_tabla1.txt
```

Esta utilería permite el uso de la cláusula WHERE y seleccionar solamente los registros que quiere vaciar en el archivo:

```
bin/mysqldump -p -where=”id_ped>200” bd_muestra pedidos > vaciado_especial.txt
```

Motivos por los cuales respaldar.

Los motivos para hacer un respaldo van desde un incendio o el robo hasta el daño accidental por parte de un usuario. También es importante la manera de restaurar los datos, en caso de un desastre, en el sentido de que sea rápida y segura y que además devuelva el funcionamiento normal a la empresa con un mínimo de retrasos.

1.7.-BUENAS PRÁCTICAS EN LA FUNCIÓN DE LA ADMINISTRACIÓN.

1.7.1 Auditoria Informática.

A finales del siglo XX, los Sistemas de TI (Tecnología de Información) han constituido las herramientas más poderosas para cualquier organización, puesto que apoyan la toma de decisiones, generando un alto grado de dependencia, así como una elevada inversión en TI.

Debido a la importancia que tienen los Sistemas de TI en el funcionamiento de una organización, existe la Auditoria Informática.

Los auditores han sido catalogados a través del tiempo como personajes siniestros que se dedican a identificar todo lo que esté mal, para denunciarlo y alertar a quien deba ser alertado. En general, el rol es percibido como una especie de representante de la inquisición dentro de la organización.

Hoy día, se debe pensar en el auditor como un elemento imprescindible para una sana operación de las instituciones. Su papel ha pasado de ser un detector de problemas, a un agente de cambio, identificador de oportunidades y emisor de propuestas de valor, su compromiso profesional va más allá de fungir como un mecanismo detectivo.

Actualmente es un asesor de negocios que brinda soluciones adecuadas al entorno y la situación interna de la organización con el fin de que ésta logre sus objetivos estratégicos.

Al igual que las demás áreas de la organización, las bases de datos deben estar sometidos a controles, por las siguientes razones: las computadoras y los centros de procesamiento de datos son blancos apetecibles para el espionaje, la delincuencia y el terrorismo. Al perder de vista la naturaleza y calidad de los datos de entrada a los Sistemas de TI se genera información errónea, con la posibilidad de que se provoque un efecto cascada y afecte a otras aplicaciones. Un Sistema de TI mal diseñado puede convertirse en una herramienta muy peligrosa para la gestión y la organización de la organización.

La auditoria informática puede ser definida como:

Un proceso evolutivo que mediante técnicas y procedimientos aplicados en una organización por personal independiente a la operación de la misma, evalúa la función de tecnología de información y su aportación al cumplimiento de los objetivos institucionales; emite una opinión al respecto y efectúa recomendaciones para mejorar el nivel de apoyo al cumplimiento de dichos objetivos.

Beneficios de la Auditoria Informática

a) Mejora la imagen pública.

Es una manifestación ante los involucrados de la organización (propietarios, clientes, proveedores, empleados, dependencias normativas) de que hay una ocupación constante en mejorar la gestión de los recursos de la organización.

b) Generación de confianza en los usuarios sobre la seguridad y control de los servicios de TI.

Uno de los atributos de la información que dependen de la percepción del usuario es su confiabilidad; el usuario tendrá confianza en la información en la medida en que confíe en la fuente de información y la infraestructura relacionada.

c) *Optimiza las relaciones internas y del clima de trabajo.*

Promueve que sean claras y compatibles las actividades de los roles involucrados en la generación, transmisión, resguardo/destrucción de la información permitiendo que cada uno de los participantes sepa qué hacer y con quien debe interactuar, evitando conflictos en los equipos de trabajo.

d) Disminución de costos de la mala calidad (reprocesos, rechazos, reclamos, entre otros).

Al observar a la información y la tecnología relacionada como activos que deben ser administrados en una serie de procesos, las relaciones internas deben definirse y las características que deben guardar dichos procesos deben establecerse, por tanto, existe menor probabilidad de que hayan rechazos, reprocesos, reclamos.

e) Balance de los riesgos en TI.

La información y la tecnología relacionada con ella están expuestas a una serie de riesgos que deben ser mitigados, transferidos, o aceptados o rechazados con conocimiento de causa. El balance de los riesgos estará en función del apetito de riesgo que tenga el tomador de decisiones.

f) *Control de la inversión en un entorno de TI a menudo impredecible.*

La organización debe administrar la información que necesita para el logro de sus objetivos y para adaptarse al entorno cambiante; por tanto, las inversiones deben responder a un análisis de costo beneficio para la organización.

ISACA

Existe una asociación internacional denominada Information Systems Audit and Control Association (ISACA); comenzó en 1967 con un grupo pequeño de profesionales con actividades similares –verificar controles en los sistemas de cómputo que se estaban convirtiendo en parte crítica en las operaciones de sus organizaciones.

En 1969 se conformaron como la asociación de auditores EDP con el fin de cubrir las necesidades únicas, diversas y de alta tecnología en el naciente campo de la TI.

La misión de ISACA consiste en mejorar el reconocimiento de la profesión de auditoria y control de las TI a través de la elaboración de materiales y marcos de trabajo, así como capacitación y certificación de sus miembros a través de la fundación (Information Systems Audit and Control Foundation)

En la actualidad cuenta con más de 50,000 miembros en más de 140 países; cuenta con 170 capítulos establecidos en más de 60 países, además de que se puede consultar su página Web como se observa en la fig. 1.18.

Fig. 1.18.- Página de ISACA.

COBIT

Un marco de trabajo conocido como Objetivos de Control para la Información y la Tecnología Relacionada (Control Objectives for Information and related Technology-COBIT) sirve como guía para la buena práctica de la auditoría de las TI, emitido por el IT Governance Institute. COBIT es un marco de control o sistema de control interno que tiene el objetivo de que las TI's sean exitosas en satisfacer los requerimientos de la organización.

COBIT es un marco de control generalmente aplicable y aceptado internacionalmente como buena práctica para controles de TI.

Provee una serie de buenas prácticas a través de un marco de dominio y procesos y presenta actividades en una estructura lógica y manejable.

Las buenas prácticas de Cobit:

- Representan el consenso de expertos fuertemente enfocados en el control y menos en la ejecución.
- Ayudan a optimizar las inversiones en TI, aseguran la entrega de servicios y proveen una medida en contra de la cual juzgar cuando las cosas van mal.

Divide la TI en 34 procesos que se integran en 4 dominios y proporciona un objetivo de control de alto nivel para cada uno. Se soporta por una serie de 318 objetivos de control detallados.

- Planeación y Organización
- Adquisición e Implementación
- Entrega y Soporte
- Monitoreo y Evaluación

Considera las necesidades fiduciarias, de calidad y de seguridad de una organización proveyendo 7 criterios de información que pueden ser usados para genéricamente definir qué requiere de TI la organización.

- Eficacia
- Eficiencia
- Disponibilidad
- Integridad
- Confidencialidad
- Confiabilidad
- Cumplimiento

1.7.2 Responsabilidades del DBA.

De acuerdo al manual de revisión de los Certified Information Systems Auditor (CISA) las responsabilidades del administrador de bases de datos son:

- Custodia información de la organización.
- Define y mantiene la estructura de los datos en el sistema corporativo de BD.
- Debe comprender a la empresa, datos de usuario y las relaciones de éstos.
- Responsable de la seguridad y clasificación de la información de los datos compartidos, almacenados en los sistemas de BD.
- Responsable del diseño real, definición y mantenimiento de las BD corporativas.

- Especificar la definición física de los datos y cambiarla para su mejor desempeño.
- Seleccionar e implementar herramientas de optimización de la BD.
- Probar y evaluar las herramientas de programadores.
- Dar soporte técnico a programadores sobre estructura de la BD.
- Implementar controles de definición, acceso, actualización y concurrencia.
- Monitorear el uso, recopilar estadísticas de desempeño y ajustar la BD.
- Definir e iniciar los procedimientos de respaldo y recuperación.

1.7.3 Mejores prácticas en la administración de bases de datos.

Es crítico que la integridad y disponibilidad de la base de datos se mantenga; esto de asegura a través de los siguientes controles, con esto se realizan unas mejores prácticas de la administración de las bases de datos:

- Establecer y obligar la definición de estándares, políticas y procedimientos.
- Establecer e implementar el respaldo de datos y los procedimientos de recuperación para asegurar la disponibilidad de la base de datos.
- Establecer los niveles necesarios de controles de acceso para los elementos de datos, tablas y archivos para prevenir acceso inadvertido o no autorizado.
- Establecer controles para asegurar que sólo el personal autorizado pueda actualizar la base de datos.
- Establecer controles para manejar los problemas de concurrencias, tales como múltiples usuarios deseando actualizar los mismos elementos de datos al mismo tiempo (por ejemplo, el uso de commit, el bloqueo de registros o archivos).
- Establecer controles para asegurar la precisión, totalidad y consistencia de los elementos de datos y las relaciones en las bases de datos. Es importante que estos controles estén contenidos en las definiciones de tablas o columnas.
- Usar puntos de verificación de las bases de datos para minimizar la pérdida de datos y los esfuerzos de recuperación para reiniciar el proceso después de una falla del sistema.
- Ejecutar la optimización de las bases de datos para reducir espacio de disco no utilizado y verificar las relaciones de datos definidas.
- Seguir los procedimientos de reestructuración cuando se hagan cambios lógicos, físicos y de procedimientos.
- Usar herramientas de reporte de desempeño de las bases de datos para monitorear y mantener la eficiencia de la base de datos.
- Minimizar la habilidad para usar medios que no sean sistemas de aplicación o fuera de los procedimientos de seguridad para acceder a los datos de la base.

Para la revisión de bases de datos deben considerarse los siguientes elementos:

Diseño. Debe verificarse:

- ✓ El modelo de base de datos existente y todas las entidades deben de tener un nombre significativo e identificarse por medio de una llave primaria o foránea.
- ✓ Las relaciones deben de tener una cardinalidad explícita y coherente.
- ✓ El modelo entidad-relación debe estar sincronizado con el esquema físico de la base de datos.
- ✓ El esquema lógico. Asegurarse de que todas las entidades existan en el diagrama entidad-relación, así como en las tablas o vistas. Todas las relaciones deben estar representadas a través de una llave primaria y una foránea, así mismo los atributos debe tener un nombre lógico y su relación especificada, no debe de aceptar valores nulos en las llaves primarias.
- ✓ El esquema físico debe revisarse, para reservar el espacio necesario de las tablas, logs y áreas temporales.

Acceso: El acceso principal a la base de datos como los procedimientos de almacenamiento y los triggers deben ser analizados. El uso de índices para minimizar el tiempo de acceso debe verificarse y realizar búsquedas, sino está basado en índices, debe existir una justificación. Si el DBMS permite la selección de los métodos o tipos de índices, se debe verificar que su uso es el correcto.

Administración: Los niveles de seguridad para todos los usuarios y sus roles deben de identificarse dentro de la base de datos, y los permisos de acceso para todos los usuarios y/o grupos de usuarios deben ser justificados. Verificar que existan procedimientos de recuperación de desastres así como de respaldos, que aseguren la confiabilidad y disponibilidad de la base de datos, así mismo, mecanismos y procedimientos que aseguren el adecuado manejo, consistencia e integridad durante los accesos concurrentes.

Interfaces: Para asegurar la integridad y confidencialidad de los datos, los procedimientos de importación y exportación de la información deben ser verificados con otros sistemas.

Portabilidad: Siempre que sea posible, debe utilizarse un Lenguaje de Consulta Estructurado (SQL - Structured Query Language).

1.8.-SEGURIDAD EN BASE DE DATOS.

La seguridad en los sistemas de información actuales se ha vuelto una parte crucial en el esquema de funcionamiento de las organizaciones. Se estima que anualmente las pérdidas ocasionadas por crímenes cibernéticos cuestan miles de dólares a nivel mundial por lo que es necesario definir e implementar medidas de seguridad en las tecnologías de información.

1.8.1 Seguridad de la información.

La seguridad en la información es de vital importancia ya que se debe garantizar que la información, la cual es la que ayuda en la toma de decisiones, esta libre de amenazas que comprometan su veracidad y por ello tomar decisiones erróneas.

Primero que nada hay que definir las amenazas que pueden perjudicar la información.

Un sistema de cómputo es aquel formado por hardware, software, medios de almacenamiento de datos, o información y personas involucradas en el mismo.

Se puede decir que un equipo de cómputo es seguro en base a su confianza, es decir, que sus componentes de hardware y software se comporten como se espera que lo hagan.

Pero existen vulnerabilidades, riesgos y amenazas las cuales son un peligro latente para los equipos de cómputo y la información.

Una *vulnerabilidad* se puede decir que es una debilidad en el sistema que puede explotarse para causar pérdida o daño en él.

Un *riesgo* es aquello que puede suceder que puede causar daño, un ejemplo de esto puede ser cuando se instala un servicio de correo no se instala un antispam o antivirus, ya que se esta corriendo el riesgo de que pueda suceder algún daño.

Mientras que una *amenaza* es cualquier circunstancia con el potencial suficiente de causar perdida o daño, por ejemplo en el mismo ejemplo del servicio de correo, una amenaza en si ya es el spam o un archivo con virus que puede llegar al correo.

Principalmente para tratar de lograr tener una buena seguridad en el sistema de información, hay que identificar los principales activos a proteger, los cuales son:

- Hardware
- Software
- Datos o información.

Existen diversos tipos de amenazas que pueden hacer que los activos sufran algún tipo de daño y con ello provocar perdidas en el sistema.

Existen 4 tipos principales de amenazas entra las cuales están:

- *Interrupción:* Un activo del sistema se pierde, es decir, se hace no disponible o inutilizable.
- *Intercepción:* Alguna parte no autorizada logra el acceso a un activo del sistema.
- *Fabricación:* Una parte no autorizada puede fabricar objetos falsos.
- *Modificación:* Cuando una parte no autorizada logra accesos y modifica el activo.

También se pueden enumerar los diferentes tipos de amenazas que existen en el medio como se pueden ver en la fig. 1.19, las cuales pueden comprometer un sistema.

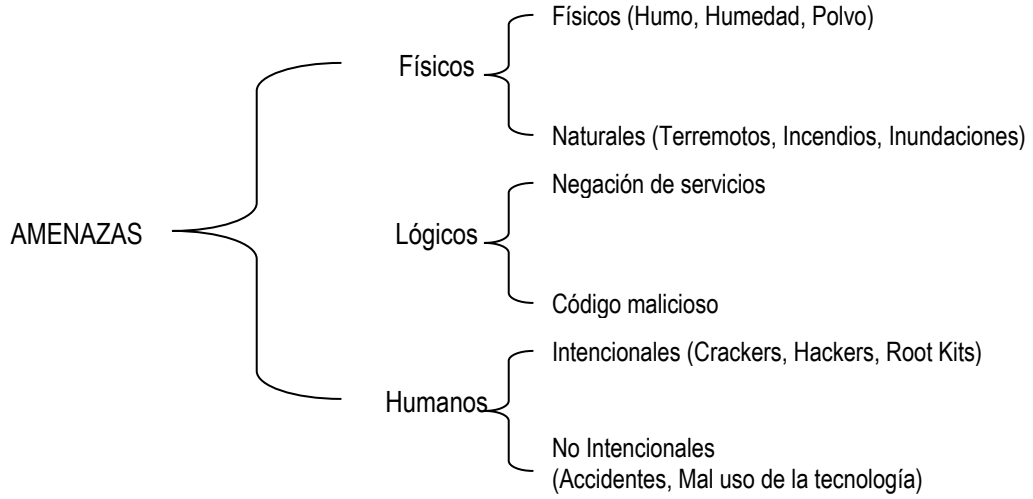


Fig. 1.19.- Amenazas que dañan al sistema.

Estas amenazas son las principales que se pueden presentar, las cuales pueden causar daños en el sistema, esto se puede realizar mediante ataques.

Un ataque es cualquier acción que explota una vulnerabilidad. Existen dos distintos tipos de ataques los cuales son:

- Ataques pasivos: Son aquellos en los que el atacante observa comportamientos o lee información, sin alterar ni el estado del sistema ni la información.

- Ataques activos: Este tipo de ataques son en los que el atacante modifica o altera ya sea el sistema o la información.

1.8.2 La seguridad en una base de datos.

La seguridad tiene que ver con las técnicas, procedimientos o medidas que reducen la vulnerabilidad del sistema.

La protección de las bases de datos es muy importante, ya que una base de datos con un bajo nivel de seguridad compromete no solamente a la base de datos misma, sino también al sistema operativo y a otros sistemas relacionados.

Los sistemas de bases de datos son sistemas extremadamente complejos y con gran dificultad para *configurar* y *asegurar*. La protección del sistema operativo y de los servicios de red en un servidor de bases de datos tiene una importancia crítica.

Los aspectos que debe cubrir la seguridad en las bases de datos para que no se vean afectados los datos contenidos en ella y por ende la información son:

- *Confidencialidad:* Los datos deben de ser protegidos de intromisión no autorizada a ellos por personas que no tienen los privilegios necesarios.
- *Integridad:* Los datos deben de ser protegidos de modificación accidental o maliciosa (considerando incluso la inserción de datos falsos o la destrucción de los mismos). Además de que el origen de los datos tiene que ser verificable.
- *Disponibilidad:* Los sistemas de bases de datos deben mantenerse operando y recuperarse en caso de perdida de datos.

Un manejador de base de datos debe de proteger sus datos de:

- Accidentes, por ejemplo errores de captura o programación.
- Uso malicioso de las bases de datos.
- Fallas del software o hardware que pudieran corromper los datos.

Algunos pasos a seguir para garantizar la seguridad de los datos y por lo tanto de la información, la cual ayuda en la toma de decisiones son:

- Autenticación de usuarios.
- Permitir acceso a los objetos de BD solamente a usuarios autorizados.
- Sistema de respaldo
- Uso de un log.
- Verificar la integridad referencial de los datos, es decir, asegurar que la relación de llave primaria y foránea se mantenga.
- Verificar que se cumplan las restricciones de los campos de las tablas.
- Verificar que las tablas tengan una llave primaria.
- Asegurarse que se cumplan los criterios de restricciones tanto en el cliente como en la BD.

- Tener un buen control de acceso discrecional, es decir, restringir el acceso a los datos u objetos a los usuarios no autorizados.
- Otorgar y revocar privilegios a los usuarios.
- Utilizar vistas para que el usuario no se meta con los datos guardados en las tablas.
- Usar un sistema de detección de intrusos.
- Cambiar las contraseñas de las cuentas creadas por omisión durante la instalación.
- Deshabilitar cuentas de invitado.
- Mantener actualizado el DBMS con las versiones más recientes del software y de los parches de seguridad.
- Impedir que las aplicaciones acepten instrucciones SQL de los usuarios y las ejecuten sobre las bases de datos.
- Revisar las bitácoras del sistema para verificar posibles intromisiones o ataques a la BD.
- Monitorear cuidadosamente los logs de error.
- Verificar los puertos del sistema, para esto se puede hacer un programa que este vigilando constantemente.
- Definir políticas de respaldo y almacenar los respaldos en lugares seguros.

1.8.3 Herramientas de apoyo a la seguridad.

Las herramientas que ayudan a mantener la seguridad son muy importantes, existe un procedimiento que ayuda a tratar de mantener la seguridad en los sistemas, este procedimiento se utiliza para tener protegido el sistema de cómputo, y el cual esta dividido en 3 diferentes pasos para garantizar esta seguridad, los cuales son:

- Actualización de Software
- Servicios
- Bitácoras

Actualización de software

Lo primero que se debe hacer es mantener las fuentes de información, como el Web, correo, etc., para estar revisando constantemente las actualizaciones del software, así como para ver las posibles vulnerabilidades del mismo, y poder instalar los parches de seguridad.

A continuación se tendra que evaluar la actualización, ya que se tiene que ver si la actualización que se va a realizar verdaderamente resuelve el problema, no sólo actualizar por actualizar, ya que eso también podría ocasionar problemas.

Después de que se evalúa la actualización y se sabe que si es viable se debe de planearla, ya que esto se deberá hacer cuando muy pocos usuarios o ningún usuario este haciendo uso del sistema, ya que si el sistema les falla los usuarios estarán inconformes, por eso es vital la planeación de la actualización.

Antes de instalar la actualización se deben hacer pruebas en un equipo o sistema que no este en producción, ya que si no se hacen estas pruebas, no se sabe como se comporta esa actualización, no solo la actualización si no con los demás componentes o software con el que interactúa, una vez que funciona correctamente en la prueba ya estará lista para aplicarla al sistema.

Una vez terminadas las pruebas y ya con la planeación hecha se procede a la instalación de la actualización para tratar que el equipo se encuentre más protegido, y se reduzca el riesgo de poder ser atacado.

Por último se debe actualizar la BD del sistema de monitoreo del sistema de archivos, para que así ya todo quede actualizado.

Servicios

En cuanto a los servicios lo primero que se debe tomar en cuenta es que servicios se van a tener abiertos en el sistema para que los usuarios autorizados puedan hacer uso del sistema, es decir se tienen que definir que servicios se tendrán abiertos.

Una vez que ya se definió que servicios son los que van a servir, se procede a cerrar los demás servicios que no se vayan a ocupar, y si por alguna razón no se sabe para que es cierto servicio que se tiene abierto, lo que se debe hacer es cerrarlo primero y después ver como afecta en el sistema, ya que si no se sabe de que se trata, tampoco se sabrá como protegerlo y ese servicio podría convertirse en una vulnerabilidad en el sistema por donde pueden atacar; si ese servicio se ocupa pues se procedería a su apertura, pero ya sabiendo para que se utiliza.

Más tarde lo que se debe considerar es la instalación de servicios de forma segura, ya que si no se hace de esta forma puede ser que se presenten complicaciones, una instalación de un servicio en forma segura podría ser por ejemplo, que si se instala un servicio de correo se deben tomar consideraciones, como lo son instalar software antispam, y tener instalado un antivirus para este servicio.

Por último lo que se debe hacer aparte de lo que ya se hizo anteriormente, es proteger estos servicios, con la ayuda de otras herramientas, como lo son firewall[47], para tener bien definido que es lo entra y sale del equipo.

Bitácoras

En cuanto a las bitácoras del sistema lo primero que se debe hacer es identificarlas, y saber donde se encuentran, ya que la mayoría de las veces estas pasan desapercibidas, porque ni siquiera se sabe en donde se encuentran, y en éstas se guardan todas las cosas que están pasando en el sistema, es por eso que son de suma importancia identificarlas.

Una vez que se identifican se debe dar a la tarea de protegerlas, ya que son muy importantes, una forma de protegerlas es hacer respaldos de las mismas, para así comparar los respaldos con las que se tienen en el sistema, ya que si se presenta algún problema en el sistema se registra en las bitácoras, y con éstas se pueden saber desde cuando se empezó a suscitar dicho problema, y así poder resolverlo de una mejor manera. Además de que si algún intruso o persona no autorizada modifica estas bitácoras del sistema con los respaldos se podría darse cuenta.

Otras herramientas que pueden ayudar a proteger la seguridad en el sistema son los escaneadores[48], uno ejemplo de estos seria Nessus.

[47] Un firewall es un dispositivo que funciona como cortafuegos entre redes, permitiendo o denegando las transmisiones de una red a la otra. Un uso típico es situarlo entre una red local y la red Internet, como dispositivo de seguridad para evitar que los intrusos puedan acceder a información confidencial.

[48] Un escaneador es una herramienta de seguridad la cual te permite escanear ya sea puertos, o archivos de la computadora.

Nessus

Esta herramienta es capaz de escanear el sistema y con esto determinar las vulnerabilidades por las cuales se puede ser atacados, aparte de lo anterior provee las posibles soluciones a las vulnerabilidades encontradas en el sistema, además de que crea reportes del estado del sistema los cuales son muy útiles.

Otra herramienta importante que puede ayudar a mantener la seguridad es un software llamado Aide.

Aide

Esta herramienta es muy útil para estar monitoreando si existen cambios en los archivos, como ya se menciono los archivos que más se deben estar monitoreando son las bitácoras las cuales son las que proveen información relevante del sistema, tanto fallos como quien ingreso al sistema.

Esta herramienta ayuda a verificar si no hubo cambios en permisos, tamaño de los archivos, así como de que si ha habido cambios en los usuarios del sistema.

Una contraseña fuerte se considera aquella que es de 10 o mas caracteres, además de que estos caracteres estén compuestos por mayúsculas, minúsculas, números y símbolos extraños.

Para esto el administrador del equipo puede auxiliarse de una herramienta llamada John de Ripper.

John de Ripper

Con esta herramienta se tiene la posibilidad de ver si los usuarios están utilizando contraseñas fuertes las cuales garanticen o reduzcan el riesgo en el sistema, ya que si usan contraseñas débiles se puede ser victimas de ataques ya que el atacante podrá explotar esta vulnerabilidad. Y si por ejemplo en las políticas del sistema se encuentra la utilización de contraseñas fuertes, con esta herramienta se verifica si en verdad se esta llevando acabo esta política por los usuarios, y de lo contrario se le informa al usuario en forma de aviso para que tome las medidas pertinentes y si reincide se aplican las sanciones correspondientes.

1.9.-PERFORMANCE AND TUNNING

Es importante el performance ya que éste indica el rendimiento, lo primero que se debe de hacer es verificar la configuración de memoria, disco, procesador.

Se debe tener un buen performance en el servidor para garantizar un buen funcionamiento en el servidor y se debe hacer uso del tunning para tratar de que el servidor se modifique para que tenga un rendimiento óptimo.

Los parámetros de configuración son estáticos o dinámicos. Los parámetros dinámicos afectan inmediatamente después de ejecutarse sp_configure.

Los parámetros estáticos requieren que el servidor de base de datos reasigne memoria. Al cambiar los parámetros estáticos se requiere reinicializar el servidor de base de datos.

El procedimiento almacenado que ayuda al proceso de performance and tunning es "*sp_configure*", y existen muchos parámetros que se pueden modificar ya sea por necesidad, es decir, cuando se requiere que el servidor acepte más usuarios para que se puedan conectar, o para hacer que el servidor de base de datos tenga un rendimiento más adecuado.

Los parámetros de configuración son valores definibles por el usuario los cuales controlan diversos aspectos del servidor de BD. Cuando se instala el servidor se tienen valores predeterminados para cada uno de estos parámetros de configuración.

El procedimiento almacenado sp_configure nos ayuda a configurar el servidor, y al ejecutarse así sin ningún parámetro muestra una lista de parámetros de configuración por grupos, sus valores actuales y predeterminados, el valor con que se han definido recientemente y la cantidad de memoria que utiliza ese valor en particular.

Algunos ejemplos de utilización de este procedimiento almacenado son:

```
> sp_configure “number of device”, “20”
```

El parámetro que modificamos con esto es el número de dispositivos que se pueden crear, ahora se pueden crear 20 dispositivos, ya que por default sólo deja crear 10 y se pueden crear como máximo 255.

```
> sp_configure “number of user connections”,”50”
```

El parámetro anterior modifica el número de usuarios conectados al mismo tiempo, es decir que a partir de ahora deje conectar a 50 usuarios simultáneamente.

Aunque estos parámetros que modificamos reducen la memoria de los datos, le afecta a *procedure cache y default data cache*, y por lo tanto hay que aumentarla para dejarla en un límite que se definió, esto se hace para que el sistema trabaje de una forma más eficiente y eso se hace modificando el parámetro total memory y el tamaño de la memoria se expresa en paginas.

```
> sp_configure “total memory”,15862
```

Las configuraciones de parámetros se guardan en dos tablas, las cuales están en master y estas tablas son: *sysconfigures y syscorconfigs*.

En la primera se guarda el valor de configure y en la segunda el valor que esta corriendo.

Así como se pueden configurar estos parámetros existen mucho más que se pueden configurar, y cuando se configuran esos parámetros el servidor de base de datos dice que si estos parámetros son dinámicos o estáticos.

1.10.-MODELADO ORIENTADO A OBJETOS

Primero que nada se debe de definir que es el paradigma[49] orientado a objetos, ya que este paradigma es la base del modelado orientado a objetos.

También se sabe que un modelo es una representación de algo real en forma física, gráfica o matemática, el cual ayuda a captar los aspectos importantes del objeto a idea a modelar.

[49] Un paradigma es un conjunto de reglas que "rigen" una determinada disciplina. Están "reglas" se asumen normalmente como "verdades incuestionables", porque son "tan evidentes" que se tornan transparentes para los que están inmersos en ellas.

Es necesario modelar ya que divide el problema en problemas más simples, además que permite ver a el problema desde varias perspectivas, por lo tanto ayuda a entender y ver la dimensión del problema.

El paradigma orientado a objetos es una filosofía para el desarrollo de sistemas basado en el modelado del mundo real a través de la identificación de objetos. Surgió inicialmente como un enfoque para la programación pero se ha extendido a todo el ciclo de desarrollo de sistemas.

Aplicaciones del paradigma orientado a objetos

- La metodología orientada a objetos facilita el diseño e implementación de sistemas con Interfaces Gráficas de Usuario (GUI)[50].
- Los métodos orientados a objetos permiten desarrollar sistemas inmersos y de tiempo real con mayor calidad y flexibilidad.
- Los sistemas de comercio electrónico requieren robustez y estabilidad que la tecnología de objetos proporciona (JAVA).

1.10.1. Metodologías orientadas a objetos.

La metodología orientada a objetos (OO), tiene como objetivo el poder ayudar a resolver problemas a través del uso del paradigma orientado a objetos, y esto lo logra auxiliándose de una de las herramientas más completas para utilizar este tipo de modelado, el cual es UML, pero antes se deben conocer algunos otros conceptos de el paradigma orientado a objetos.

Como ya se menciono el paradigma OO es una filosofía que ayuda a modelar el mundo real a través de la identificación de los objetos. Ya que un objeto es la parte más importante de este paradigma, ya que todo lo ve como objeto.

Objeto

Representa un elemento que tiene atributos y realiza un conjunto de acciones. Se puede decir que un objeto es algo que tiene:

- Estado
- Comportamiento
- Identidad

Otros aspectos importantes en el paradigma OO son los siguientes conceptos:

- *Clase:* Conjunto de objetos que comparten una estructura en común y un comportamiento en común.
- Abstracción: Tomar sólo las propiedades del objeto que interesan, es decir, representa las características esenciales del objeto.
- Polimorfismo: Permite realizar la misma operación de forma diferente para cada objeto.
- Herencia: Relación para rehusar clases existentes y así definir nuevas. Las clases hijas conservan la estructura y el comportamiento de la clase padre.
- Encapsulamiento: Propiedad que permite asegurar que el contenido de la información de un objeto está oculta al mundo exterior, así que se puede decir que protege datos.
- Mensajes: Es la forma de comunicación entre objetos, ya que un objeto envía a otro un mensaje para realizar una operación y el objeto receptor ejecuta la operación.

[50] GUI o Interfaz Grafica del Usuario es el medio por el cual se comunica el usuario con la computadora.

1.10.2 Análisis orientado a objetos.

Para realizar el análisis orientado a objetos, hay que auxiliarse de una herramienta llamada UML (Unified Modeling Language), la cual ayuda a visualizar, especificar y documentar cada una de las partes que comprende el desarrollo del software.

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. Se ha convertido en el estándar de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh, utilizando métodos de otros. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE.

- Lenguaje. Mediante la notación permite expresar y comunicar conocimiento.
- Unificado. Integra lo mejor de varios autores, notaciones y técnicas.
- Modelado. Permite representar de manera abstracta aspectos reales.

El diagrama que se muestra en la fig. 1.20, nos muestra las técnicas más importantes que se unieron para poder lograr UML.

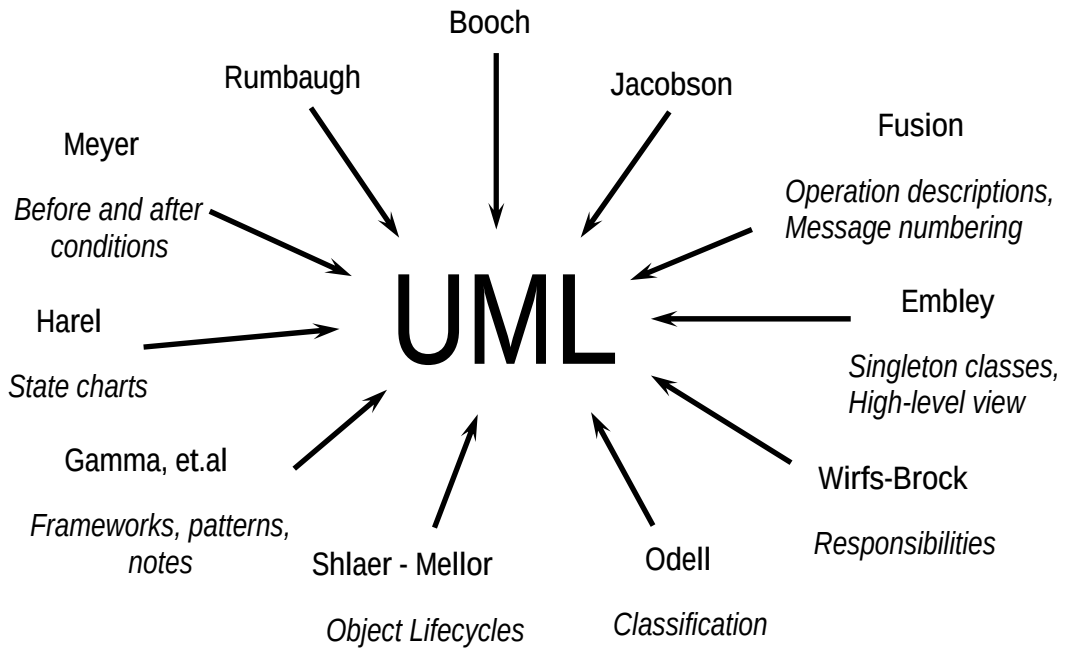


Fig. 1.20.- Técnicas que forman UML.

Ventajas del UML

- Define una notación expresiva y consistente.
- Facilita la comunicación con otros.
- Permite detectar omisiones o inconsistencias.
- Es aplicable a sistemas sencillos y complejos.
- Es un estándar en la industria de construcción de software.
- Existen herramientas en el mercado para modelar y generar código a partir de UML.

- Provee beneficios significativos para los ingenieros de software y las organizaciones al ayudarles a construir modelos rigurosos, trazables y fáciles de mantener, que soporten el ciclo de vida de desarrollo de software completo.

Diagramas de UML

UML incluye una serie de diagramas; especifica la notación para representarlos pero no describe cómo crearlos. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representan la arquitectura del proyecto. Cada diagrama usa la notación pertinente y la suma de estos diagramas crean las diferentes vistas.

UML introduce nuevos diagramas que representa una visión dinámica del sistema. Es decir, gracias al diseño de la parte dinámica del sistema se puede dar cuenta en la fase de diseño de problemas en la estructura al propagar errores o de las partes que necesitan ser sincronizadas, así como del estado de cada una de las instancias en cada momento.

Vistas existentes en UML

- *Vista casos de uso:* Se forma con los diagramas de casos de uso, colaboración, estados y actividades.
- *Vista de diseño:* Se forma con los diagramas de clases, objetos, colaboración, estados y actividades.
- *Vista de procesos:* Se forma con los diagramas de la vista de diseño. Recalcando las clases y objetos referentes a procesos.
- *Vista de implementación:* Se forma con los diagramas de componentes, colaboración, estados y actividades.
- *Vista de despliegue:* Se forma con los diagramas de despliegue, interacción, estados y actividades.

Se dispone de dos tipos diferentes de diagramas los que dan una vista estática del sistema y los que dan una visión dinámica.

Los diagramas estáticos son:

- *Diagrama de clases:* Muestra las clases, interfaces, colaboraciones y sus relaciones. Son los más comunes y dan una vista estática del proyecto.
- *Diagrama de objetos:* Es un diagrama de instancias de las clases mostradas en el diagrama de clases. Muestra las instancias y como se relacionan entre ellas. Se da una visión de casos reales.
- *Diagrama de componentes:* Muestran la organización de los componentes del sistema. Un componente se corresponde con una o varias clases, interfaces o colaboraciones.
- *Diagrama de despliegue:* Muestra los nodos y sus relaciones. Un nodo es un conjunto de componentes. Se utiliza para reducir la complejidad de los diagramas de clases y componentes de un gran sistema.
- *Diagrama de casos de uso:* Muestran los casos de uso, actores y sus relaciones. Muestra quien puede hacer que y relaciones existen entre acciones (casos de uso). Son muy importantes para modelar y organizar el comportamiento del sistema.

Los diagramas dinámicos son:

- *Diagrama de secuencia (Diagrama de colaboración):* Muestran a los diferentes objetos y las relaciones que pueden tener entre ellos, los mensajes que se envían entre ellos. Son dos diagramas diferentes, que se puede pasar de uno a otro sin perdida de información, pero que dan puntos de vista diferentes del sistema.

- *Diagrama de estados:* muestra los estados, eventos, transiciones y actividades de los diferentes objetos. Son útiles en sistemas que reaccionen a eventos.

- *Diagrama de actividades:* Es un caso especial del diagrama de estados. Muestra el flujo entre los objetos. Se utilizan para modelar el funcionamiento del sistema y el flujo de control entre objetos.

Diagrama de Casos de Uso

El diagrama de Casos de Uso captura el comportamiento de un sistema. Los Casos de Uso representan toda la funcionalidad del sistema. El modelo de Casos de Uso es una especificación completa de todas las formas posibles de utilizar un sistema: requisitos funcionales.

Básicamente un diagrama de Casos de Uso:

- Describe los servicios que el sistema proporciona al usuario.
- Proporciona información acerca de los usuarios (actores) del sistema (Actores: Humanos, otros sistemas, máquinas).
- Muestra la naturaleza de las interacciones entre el actor y el sistema (casos de uso), es decir, relaciona actores y casos de uso.

Para realizar un diagrama de Casos de Uso se requieren los siguientes componentes:

- Actor
- Caso de Uso
- Relación

Actor

Un Actor es un usuario del sistema, que necesita o usa algunos de los casos de uso. Se representa mediante un personaje, acompañado de un nombre significativo, si es necesario.

Los actores son los que intercambian información con el sistema, receptor pasivo de información, puede ser una persona, una máquina u otro sistema, y se representa por un muñequito.

Caso de Uso

Representado por una elipse, tiene un nombre que indica su funcionalidad, este interactúa con el actor. Un Caso de Uso es una interacción típica entre un usuario (Actor) y un sistema.

Relación

Es la interacción que existe entre un actor y un caso de uso, denota la participación del actor en el caso de uso determinado. Se tienen las relaciones normales que permiten comunicar al actor con el caso de uso y dos más:

- *Uses:* Cuando se tiene una porción de comportamiento que es similar más de un Caso y no se quiere duplicar la descripción.
- *Extend:* Se utiliza cuando se tiene un Caso de Uso que incorpora dos o más escenarios con diferencias significativas, así se muestra con un Caso de Uso principal y uno o más secundarios.

Un ejemplo de este diagrama se muestra en la fig.1.21, en donde se realiza una transferencia bancaria.

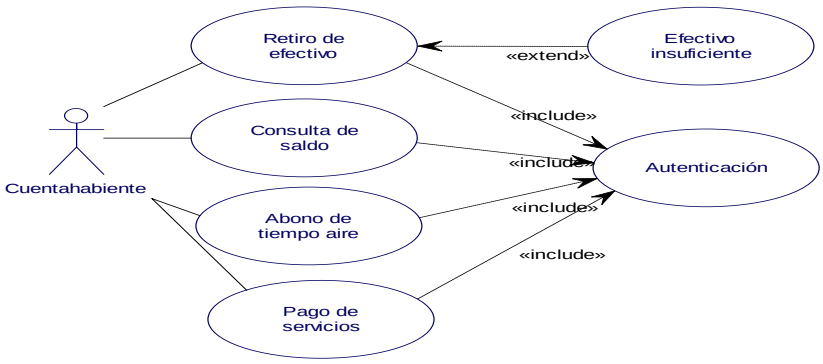


Fig. 1.21.-Diagrama de caso de uso.

Diagrama de actividades

Un diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior.

La diferencia entre los diagramas de flujo y los diagramas de actividad es que pueden ser representadas actividades en paralelo. Esto es importante para el modelado de negocios, cuando los procesos no son secuenciales y pueden ser representados de forma paralela.

Sus componentes principales son:

- *Actividad:* Una actividad es una acción a realizar. Se representa mediante un rectángulo con las orillas semi-onduladas.
- *Barra de sincronización:* La barra de sincronización permite iniciar acciones una vez que se han realizado actividades concurrentes.
- *Decisión:* La decisión es un punto en el que se pueden seguir alternativas distintas de acuerdo al resultado de la actividad anterior.
- *Condición de guarda:* Son los posibles resultados de una acción que servirán como condición para la realización de otra.
- *Estado inicial:* Es el inicio del diagrama, además de ser obligatorio y sólo se permite uno en cada diagrama, y se representa por un círculo sólido.
- *Estado final:* Es representado por un ojo, esté es el final del diagrama y puede existir más de uno.

Un ejemplo que representa el uso de esté diagrama es el de la fig. 1.22.

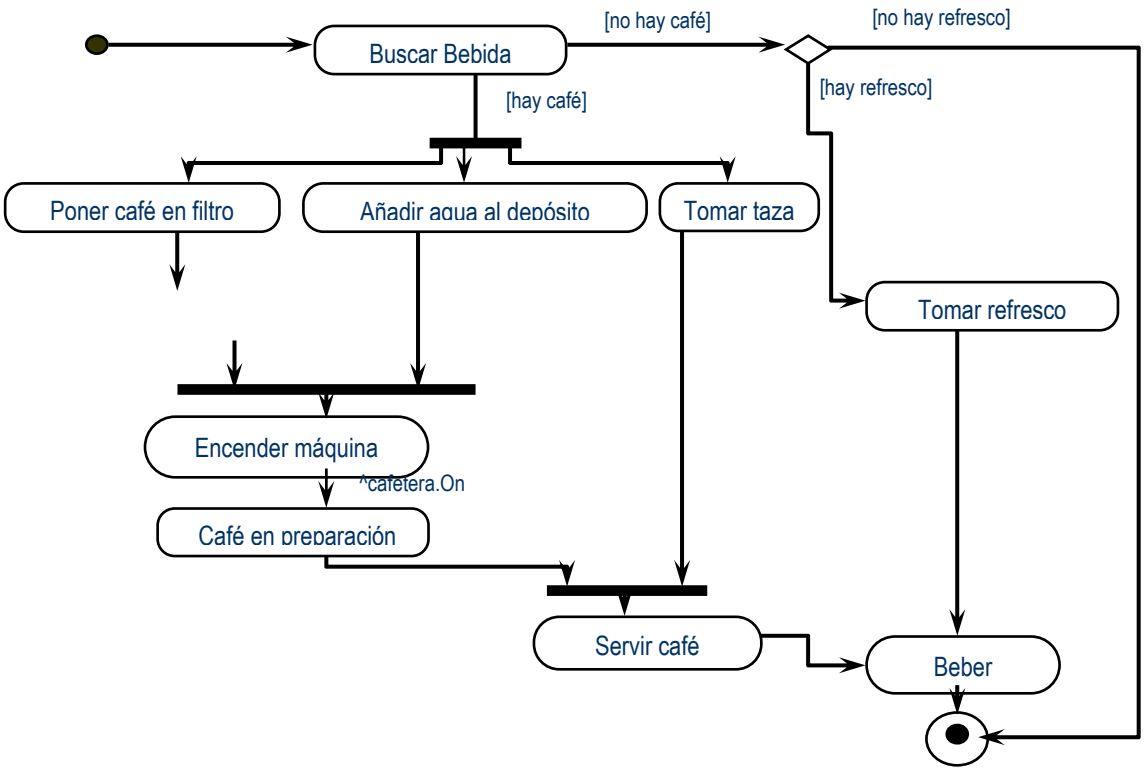


Fig. 1.22.- Diagrama de actividades.

Diagrama de secuencias

Es uno de los dos tipos de diagramas de interacción. Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes, como también muestra el uso de los mensajes de las clases diseñadas en el contexto de una operación.

Sus principales componentes son:

- *Línea de vida de un objeto:* Es representada con una línea punteada con un rectángulo de encabezado, el nombre se debe encontrar dentro del rectángulo.
- *Activación*: Muestra el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna operación. Se denota como un rectángulo delgado sobre la línea de vida del objeto.
- *Mensaje:* Se denota como una línea sólida dirigida, desde el objeto que emite el mensaje hasta el objeto que lo ejecuta.
- *Notas:* Las notas pueden agregarse para dar más información al diagrama.

En la fig. 1.23 se muestra un esquema general de cómo se representa un diagrama de secuencias.

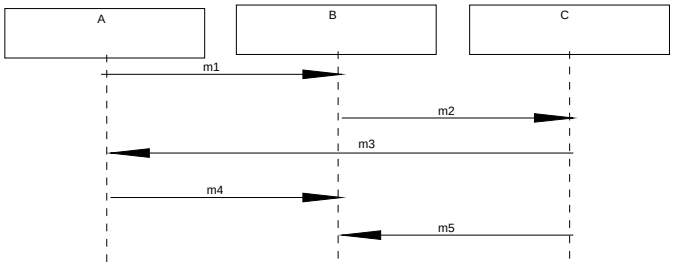


Fig. 1.23.-Esquema de un diagrama de secuencias.

Diagrama de Colaboración

Es el otro tipo de diagrama de interacción. Un diagrama de colaboración es una forma de representar interacción entre objetos, alterna al diagrama de secuencia. A diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación, y ciclos en la ejecución.

Los elementos que intervienen en estos diagramas son:

- *Objetos*: Un objeto se representa con un rectángulo, que contiene el nombre y la clase del objeto. Un objeto es una instancia de una clase que participa como una interacción, existen objetos simples y complejos. Un objeto es activo si es capaz de iniciar la actividad de control, mientras que un objeto es pasivo si mantiene datos pero no inicia la actividad.
- *Ligas entre objetos*: Un enlace es una instancia de una asociación en un diagrama de clases. Se representa como una línea continua que une a dos objetos en este diagrama. Esta acompañada por un número que indica el orden dentro de la interacción y por un estereotipo que indica que tipo de objeto recibe el mensaje. El enlace puede ser reflexivo si conecta a un elemento consigo mismo. La existencia de un enlace entre dos objetos indica que puede existir un intercambio de mensajes entre los objetos conectados.
- *Mensajes intercambiados entre objetos:* Expresa el envío de un mensaje. Se representa mediante una flecha dirigida cercana a un enlace. Durante la ejecución de un diagrama de colaboración se crean y destruyen objetos y enlaces.

Un problema que podría ilustrar muy bien este diagrama es la inscripción de un alumno en sus materias para un ciclo escolar como se aprecia en la fig. 1.24.

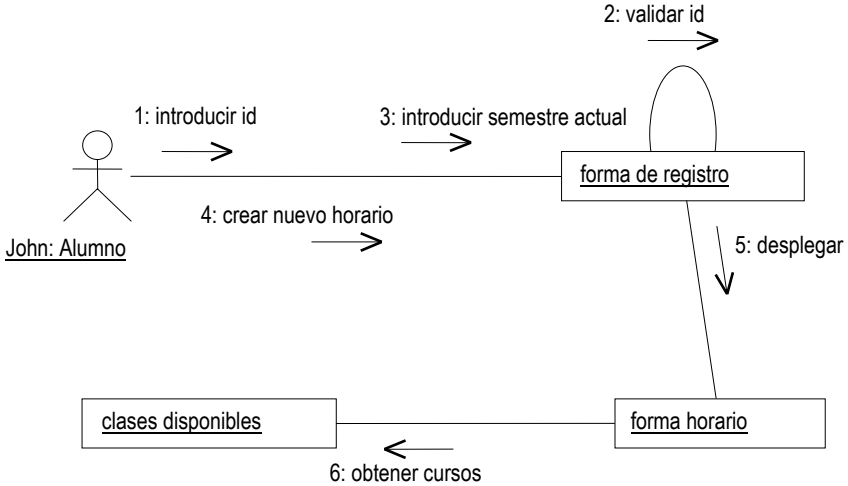


Fig. 1.24.- Diagrama de colaboración.

Diagramas de Estado

Representan la secuencia de estados por los que un objeto o una interacción entre objetos pasan durante su tiempo de vida en respuesta a estímulos recibidos. Representa lo que se puede denominar en conjunto una máquina de estados.

Un estado en UML es cuando un objeto o una interacción, satisface una condición, desarrolla alguna acción o se encuentra esperando un evento.

Un diagrama de transición de estado se usa para mostrar la historia de vida de una clase dada, los eventos que causan una transición de un estado a otro, y las acciones que resultan de un cambio de estado.

Sus principales componentes de este diagrama son:

- *Estado:* Se representa como un rectángulo redondeado. Un estado es una de las condiciones posibles en las que puede existir un objeto.
- *Evento:* Es la ocurrencia de alguna situación que sucede en un punto del tiempo; tiene una localización en tiempo y espacio y no tiene duración.
- *Transición:* Una transición es un cambio de un estado original a un estado sucesor como resultado de algunos estímulos.
- *Condición de guarda:* Es una expresión booleana de valores de atributos que permiten una transición sólo si la condición es verdadera.
- *Acción:* Es una operación que se asocia a una transición, toma una cantidad de tiempo para completarse, se considera no interrumpible.
- *Actividad:* Es una operación que toma tiempo para completarse. Las actividades se asocian con un estado o una actividad inicia cuando se introduce el estado, puede ejecutarse hasta el final o puede ser interrumpido por una transición que sale.

La fig. 1.25 muestra el uso del diagrama de estados.

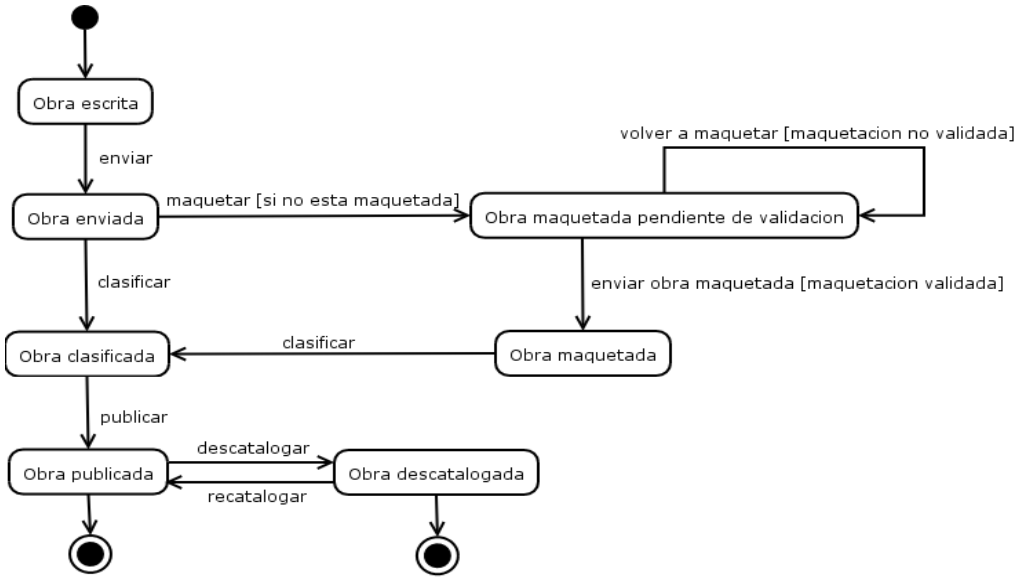


Fig. 1.25.-Diagrama de estados.

Diagramas de Componentes

El Diagrama de Componentes se usa para modelar la estructura del software, incluyendo las dependencias entre los componentes de software, los componentes de código binario, y los componentes ejecutables. En el Diagrama de Componentes modela componentes del sistema, a veces agrupados por paquetes, y las dependencias que existen entre componentes (y paquetes de componentes), es decir, se utilizan para modelar la vista estática de un sistema.

Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. En él se sitúan librerías, tablas archivos, ejecutables y documentos que formen parte del sistema.

Uno de los usos principales es que puede servir para ver que componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. Una representación de este diagrama es mostrada en la fig. 1.26.

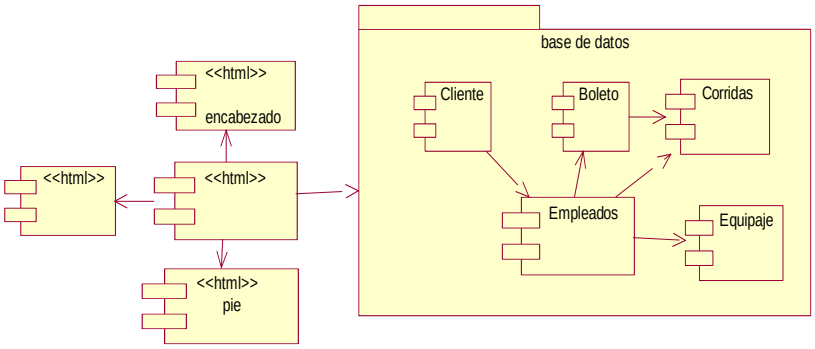


Fig. 1.26.- Diagrama de componentes.

Diagramas de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Es decir se sitúa el software en el hardware que lo contiene. Cada hardware se representa como un nodo.

Un nodo se representa como un cubo, es un elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes. Un ejemplo de este tipo de diagramas se ve en la fig. 1.27.

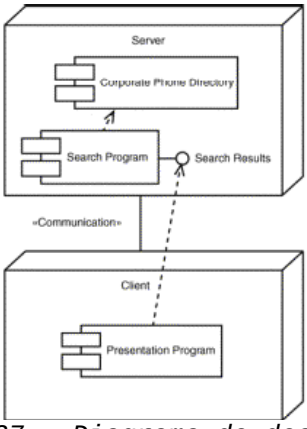


Fig. 1.27.- Diagrama de despliegue.

Diagrama de Clases

Se utilizan para modelar la vista de diseño estática de un sistema. Esta vista soporta principalmente los requisitos funcionales de un sistema, es decir, los servicios que el sistema debe proporcionar a sus usuarios finales.
En el diagrama de clases es donde se definen las características de cada una de las clases, interfaces, colaboraciones y relaciones de dependencia y generalización. Es decir, es donde se da rienda suelta a los conocimientos de diseño orientado a objetos, definiendo las clases e implementando las ya típicas relaciones de herencia y agregación. En el diagrama de clases se debe definir a éstas y a sus relaciones.

Los diagramas de clases se utilizarán de una de tres formas:

1. *Para modelar el vocabulario de un sistema:* El modelado del vocabulario de un sistema implica tomar decisiones sobre qué abstracciones son parte del sistema en consideración y cuáles caen fuera de sus límites. Los diagramas de clases se utilizan para especificar estas abstracciones y sus responsabilidades.

2. *Para modelar colaboraciones simples:* Una colaboración es una sociedad de clases, interfaces y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de todos los elementos.

3. *Para modelar un esquema lógico de base de datos.*

Un ejemplo de un diagrama de clases es el de la fig. 1.28, utilizado para representar la venta de boletos de autobús.

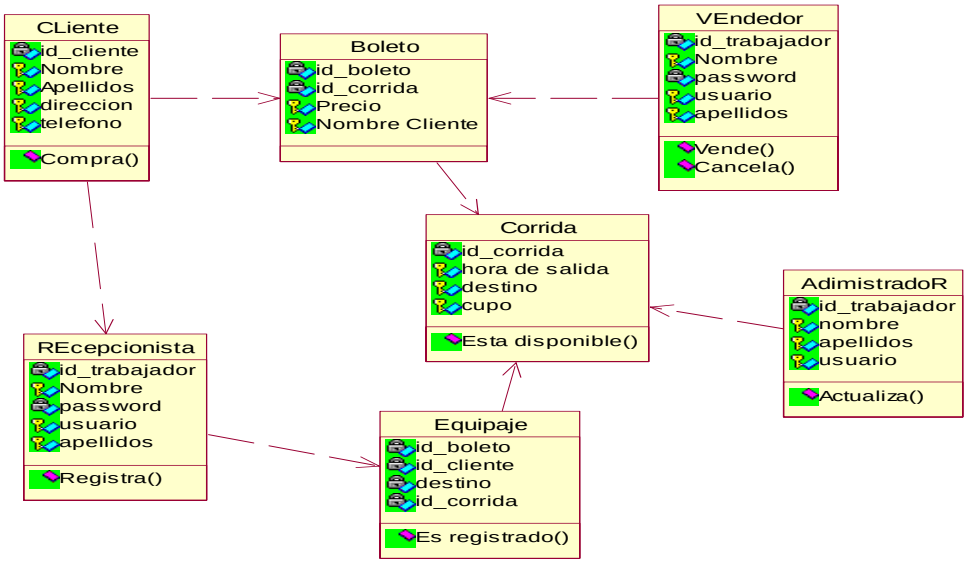


Fig. 1.28.- Diagrama de clases.

1.10.3 Principales características de los Manejadores de base de datos orientados a objetos.

Características de los manejadores de base de datos OO

- Permiten el almacenamiento de estructuras datos complejas que no pueden ser almacenadas fácilmente en bases de datos convencionales.
- No sólo almacenan tablas, columnas y renglones.
- Fueron diseñadas para ser integradas de forma transparente con la programación OO.
- Almacenan objetos; no son renglones y columnas.
- Permiten manejar datos complejos.
- Soportan el paradigma OO
- Existencia de un identificador de objetos (OID), el cual nunca es manipulado por el usuario, ya que el DBMS es el único que manipula y asigna este identificador.
- Un OODB permite que los objetos tengan atributos de conjuntos de objetos por lo que las relaciones N:N pueden ser representadas directamente sin necesidad de entidades transitivas.

Desventajas

- Complejidad.
- Inmadurez.
- Falta de estandarización.
- Falta de familiaridad.
- La mayoría de los desarrolladores domina RDBMS
- Es más fácil utilizar lo que ya se conoce.
- No se recomiendan cuando se tienen pocos datos y éstos son muy simples (no son eficientes)
- No se recomiendan cuando no se está usando un lenguaje OO
- La mayoría de las herramientas está orientada a RDBMS.
- La mayoría de las empresas que las venden son relativamente pequeñas.

1.11.-TÓPICOS AVANZADOS DE BASE DE DATOS.

1.11.1. Minería de datos.

El Data Mining (DM) o minería de datos es un mecanismo de explotación, consistente en la búsqueda de información valiosa en grandes volúmenes de datos. Está muy ligada a los Data Warehousing[51] que proporcionan la información histórica con la cual los algoritmos de minería de datos tienen la información necesaria para la toma de decisiones.

Etapas principales del proceso de Data Mining:

1. Determinación de los objetivos: Delimitar los objetivos que el cliente desea bajo la orientación del especialista en Data Mining.

[51] Un data warehousing es una bodega donde están almacenados todos los datos para la gestión de la empresa.

2. Preprocesamiento de los datos: Se refiere a la selección, la limpieza, el enriquecimiento, la reducción y la transformación de las bases de datos. Esta etapa consume generalmente alrededor del setenta por ciento del tiempo total de un proyecto de Data Mining.

3. Determinación del modelo: Se comienza realizando un análisis estadístico de los datos, y después se lleva a cabo una visualización gráfica de los mismos para tener una primera aproximación. Según los objetivos planteados y la tarea que debe llevarse a cabo, pueden utilizarse algoritmos desarrollados en diferentes áreas de la Inteligencia Artificial[52].

4. Análisis de los resultados: verifica si los resultados obtenidos son coherentes y los coteja con los obtenidos por el análisis estadístico y de visualización gráfica. El cliente determina si son novedosos y si le aportan un nuevo conocimiento que le permita considerar sus decisiones.

Las técnicas de DM son el resultado de un largo proceso de investigación y desarrollo de productos. Esta evolución comenzó cuando los datos de negocios fueron almacenados por primera vez en computadoras, y continuó con mejoras en el acceso a los datos, y más recientemente con tecnologías generadas para permitir a los usuarios navegar a través de los datos en tiempo real.

La minería de datos toma este proceso de evolución más allá del acceso y navegación de los datos, hacia la entrega de información. DM está listo para su aplicación en la comunidad de negocios porque está soportado por tres tecnologías que ya están suficientemente maduras:

- Recolección masiva de datos
- Potentes computadoras con multiprocesadores
- Algoritmos de Data Mining

Los componentes esenciales de la tecnología de DM han estado bajo desarrollo por décadas, en áreas de investigación como estadísticas, inteligencia artificial y aprendizaje de máquinas. Hoy, la madurez de estas técnicas, junto con los motores de bases de datos relacionales, hicieron que estas tecnologías fueran prácticas para los entornos de Data Warehouse actuales.

El DM envuelve muchos diferentes algoritmos para resolver diferentes tareas. El algoritmo examina los datos y determina un modelo que es cerrado. Los algoritmos de DM pueden ser caracterizados consistiendo en tres partes:

Modelo: El propósito del algoritmo es el de colocar el dato en un modelo.
Preferencia: Algún criterio debe ser usado para colocar un modelo en otro.
Buscar: Todos los algoritmos requieren alguna técnica para buscar el dato.

Modelos del Data Mining

El modelo que es creado puede ser predictivo o descriptivo.

Modelo predictivo.

Hace una predicción acerca de los valores de los datos usando la compresión de los resultados encontrados desde diferentes datos. Un modelo predictivo puede hacerse basado sobre el uso de otro historial de datos.

[52] La Inteligencia Artificial es la rama de la informática que desarrolla procesos que imitan a la inteligencia de los seres vivos.

Modelo descriptivo.

Éste identifica patrones o relaciones en los datos. A diferencia del modelo predictivo, sirve como un camino para explorar las propiedades de los datos examinados, no para predecir nuevas propiedades.

1.11.2. Data Warehousing.

El DataWarehouse (DW) no es un producto que pueda ser comprado en el mercado, sino más bien un concepto que debe ser construido. DW es una combinación de conceptos y tecnología que cambian significativamente la manera en que es entregada la información a la gente de negocios. El objetivo principal es satisfacer los requerimientos de información internos de la empresa para una mejor gestión, con eficiencia y facilidad de acceso.

El DW puede verse como una bodega donde están almacenados todos los datos necesarios para realizar las funciones de gestión de la empresa, de manera que puedan utilizarse fácilmente según se necesiten. El contenido de los datos, la organización y estructura son dirigidos a satisfacer las necesidades de información de analistas.

El DW intenta responder a la compleja necesidad de obtención de información útil sin el sacrificio del rendimiento de las aplicaciones operacionales.

Existen muchas definiciones para el DW, la más conocida fue propuesta por Inmon la cual dice lo siguiente: "*Un DW es una colección de datos integrados, enfocados en un tema, no-volátiles y que varían en el tiempo, los cuales soportan el proceso de toma de decisiones de las empresas*".

En 1993, Susan Osterfeldt publica una definición que sin duda acierta en la clave del DW: "*Yo considero al DW como algo que provee dos beneficios empresariales reales: Integración y Acceso de datos. DW elimina una gran cantidad de datos inútiles y no deseados, como también el procesamiento desde el ambiente operacional clásico*". Esta última definición refleja claramente el principal beneficio que el DW aporta a la empresa, eliminar aquellos datos que obstaculizan la labor de análisis de información y entregar la información que se requiere en la forma más apropiada, facilitando así el proceso de gestión.

Los Data Warehouse se almacenan en servidores OLAP (Procesamiento Analítico En Línea), el cual surge como un proceso para ser usado en el análisis de negocios. El DW se debe poner por separado de las bases de datos operacionales de la empresa.

DataMart.

El concepto DataMart es una extensión natural del DW, y está enfocado a un departamento o área especifica, como por ejemplo los departamentos de ventas, almacén, etc. Permitiendo así un mejor control de la información que se está abarcando.

OLAP, R-OLAP y M-OLAP.

Un sistema OLAP se puede entender como la generalización de un generador de informes. Las aplicaciones informáticas clásicas de consulta, orientadas a la toma de decisiones, deben ser programadas. Atendiendo a las necesidades del usuario, se crea una u otra interfaz. Los sistemas OLAP evitan la necesidad de desarrollar interfaces de consulta, y ofrecen un entorno único válido para el análisis de cualquier información histórica, orientado a la toma de decisiones. A cambio, es necesario definir dimensiones, jerarquías y variables, organizando de esta forma los datos.

Un DW puede implementarse sobre los diversos tipos de SMBD. Y los servidores OLAP pueden clasificarse tomando en cuenta el modelo que utilizan:

- Servidores OLAP Relacionales (R-OLAP).
- Servidores OLAP Multidimensionales (M-OLAP).

R-OLAP es la arquitectura de base de datos multidimensional en la que los datos se encuentran almacenados en una base de datos relacional, la cual tiene forma de estrella (también llamada copo de nieve o araña). En R-OLAP, en principio la base de datos sólo almacena información relativa a los datos en detalle, evitando acumulados (evitando redundancia).

En un sistema M-OLAP, en cambio, los datos se encuentran almacenados en archivos con estructura multidimensional, los cuales reservan espacio para todas las combinaciones de todos los posibles valores de todas las dimensiones de cada una de las variables, incluyendo los valores de dimensión que representan acumulados.

Para comprender mejor el funcionamiento de esta tecnología se puede apreciar en la fig. 1.29.

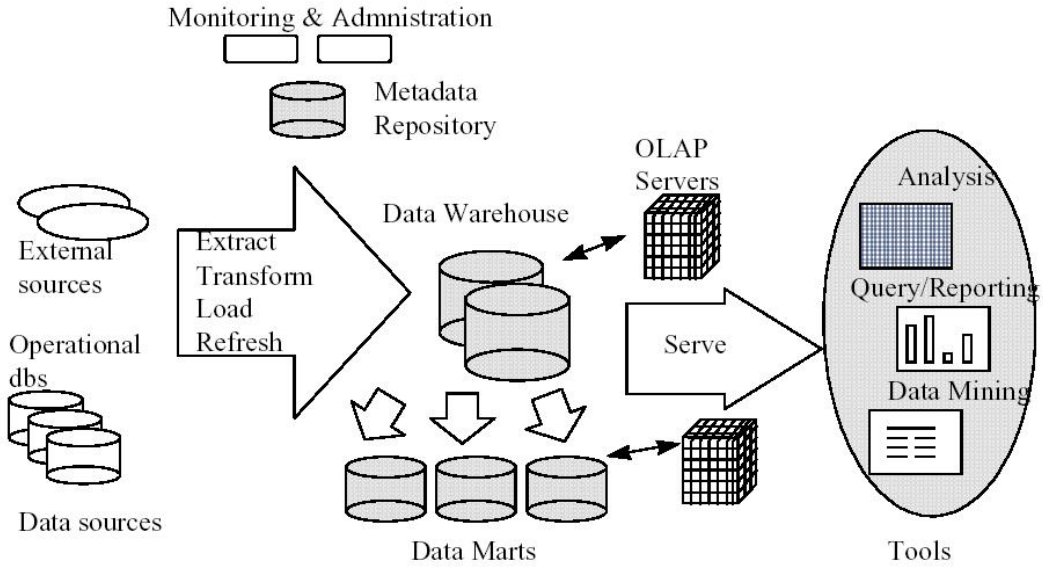


Fig. 1.29.- Arquitectura de un datawarehousing.

El presente capitulo sirvió para tener un panorama general de lo que se trata la administración de bases de datos en diferentes manejadores ya sea comerciales o de software libre, además de ver las características de los manejadores para su implementación.

Proporcionó información para comprender los objetivos fundamentales que debe tener el administrador de la base de datos, además de proporcionarle una visión mayor de otras herramientas y aplicaciones para poder desempeñar de una forma más eficaz su trabajo de resguardar la información, la cual como ya se dijo es de vital importancia para la empresa, y el que tiene la información confiable y segura tiene el poder.

CAPITULO 2.- CASO PRÁCTICO.

Como se ha visto hasta ahora el trabajo de un Administrador de Base de Datos, es muy importante para la empresa, ya que este trabaja directamente con la información con la que cuenta la empresa y en nuestros días es está la que tiene un mayor valor para las personas debido a que les ayuda a la toma de decisiones.

Por lo tanto se requiere realizar una aplicación que ayude en la administración de los usuarios que tienen acceso a esa información, así como el otorgar la propiedad de bases de datos a diferentes usuarios según estos ocupen estas para realizar su trabajo. Además se requiere que desde la aplicación que en este caso es una aplicación Web se realicen respaldos para tener una forma de preservar nuestra información.

2.1 OBJETIVO DEL SISTEMA.

El nuevo sistema Web permitirá que el administrador de base de datos, tenga una forma gráfica, intuitiva y fácil de realizar el trabajo de administración de los usuarios dentro del sistema, además de gestionar de una manera optima las bases de datos y tener un apoyo para realizar respaldos con tan solo dar un clic.

Esto es muy importante para el administrador de base de datos, ya que le ayuda a desempeñar su trabajo de una manera más fácil, rápida y sin tener que utilizar comandos para la administración dentro del manejador de base de datos, así tener mayor tiempo para desempeñar otras actividades que deber realizar como por ejemplo la puesta a punto del sistema manejador de base de datos.

2.2 REQUISITOS DEL SISTEMA.

Se requiere un sistema que cuente con una interfaz grafica de fácil manejo y accesible vía Web. Además como ya se dijo que la información es la más importante para cualquier persona, entonces esta aplicación Web deberá implementar un determinado grado de seguridad, y esto lo hará teniendo un que pasar por una pantalla de logueo antes de poder entrar directamente al sistema, esto implica que el usuario debe tener un usuario válido para poder ingresar.

Pero como se esta hablando de un sistema Web, también deberá implementar una sesión para cada usuario que este conectado al mismo, ya que si no cualquier persona se podría pasar por alto la pantalla de logueo sólo con el simple hecho de teclear la dirección Web de la página en donde se administran los usuarios, respaldos o bases de datos.

Se requiere que el sistema cuente con los siguientes módulos:

- Usuarios
  - Crear de usuarios
  - Modificar de usuarios
  - Eliminar de usuarios
  - Listar usuarios

- Bases de Datos
  - Crear bases de datos
  - Modificar bases de datos
  - Eliminar bases de datos
  - Listar base de datos

- Respaldos
  - Respaldo Total
  - Respaldo de base de datos

Además se agregara un módulo el cual será el de grupos, este módulo servirá sólo para poder agrupar los usuarios en diferentes grupos ya que a partir de estos grupos el administrador podrá otorgar privilegios a los usuarios sobre los objetos de las bases de datos, ya sea una tabla, una vista, etc., ya que con esto se le simplificara el trabajo de no estar asignando estos privilegios a los diversos usuarios y por lo tanto con tan sólo agregar estos privilegios al grupo automáticamente se le asignara a cada usuario que forme parte de este grupo.

- Grupos
  - Crear grupos
  - Eliminar grupos
  - Listar grupos

2.3 HERRAMIENTAS PARA LA CONSTRUCCIÓN DE SISTEMA.

Para la realización e implantación del sistema se requerirán los siguientes componentes de software:

- Sistema Operativo Linux
- Servidor Web Apache 1.3.37
- Manejador de base de datos Postgresql 8.1.5
- Interprete PHP 4.4.5

Además de que se utilizaran otras herramientas adicionales para la construcción de la interfaz, como lo son:

- HTML
- Flash
- Hojas de Estilo

2.4 SISTEMA AUXILIAR DE MANEJO DE BASES DE DATOS (SAMBa).

2.4.1 Entrar al sistema

Al iniciar la entrada al sistema, lo primero que aparece cuando se pone la dirección URL [53] es una pantalla como la que se muestra en la fig. 2.1, la cual muestra una imagen con el nombre del sistema y un botón que dice entrar, ambos realizados en flash[54].

Una vez que se da un clic al botón para entrar al sistema llevará a una pantalla de logueo[55] como se observa en la fig. 2.2, en donde se tendrá que teclear el login y el password de un usuario que sea válido dentro del sistema para poder ingresar a realizar las tareas de administración.

[53] URL significa *Uniform Resource Locator*, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.
[54] Flash es un software o programa que nos permite hacer imágenes animadas.

Fig. 2.1.- Pantalla de inicio del sistema.

Fig. 2.2.- Pantalla de logueo.

En este punto el sistema puede mostrar dos pantallas diferentes, ya que aquí se válida si el usuario que esta intentando ingresar al sistema es válido dentro del mismo o si no lo es. Que no sea válido en el sistema se puede deber a muchos factores, los cuales pueden ser que haya escrito el login o password incorrectos, que no tengan un usuario válido para poder ingresar, o que intenten saltarse esta pantalla de logueo para entrar al sistema sin tener que pasar por aquí, y así no tener que contar con un usuario válido, y esto lo pretenden hacer saltándose a una pantalla de administración poniendo la dirección de esta página en la barra de direcciones del browser, pero como se cuenta con sesiones esto no es posible y esto hace que se redireccione a la perdona que quiere entrar sin loguearse a la página de logueo automáticamente.

[55] Logueo es la página de autenticación donde se ingresa los datos del usuario que quiere ingresar al sistema para verificar si esos datos son validos.

Otro factor puede ser que el usuario y el password sean correctos, pero que la fecha de caducidad de la cuenta que le fue asignada ya no este vigente y por esa razón no pueda ingresar al sistema. Y por lo cual mostrara la pantalla como se muestra en la fig. 2.3 en todos los casos anteriormente descritos.

Fig. 2.3.-Pantalla de error del sistema.

Por el contrario si el sistema verifica que el login y password introducidos son válidos mostraran una pantalla de bienvenida como se muestra en la fig. 2.4, donde se podrá empezar a realizar la gestión de usuarios, bases de datos, grupos y respaldos.

En esta pantalla aparece el logo del sistema así como la información de que usuario se encuentra logueado en ese momento, una liga que ayuda a salir del sistema, la cual termina la sesión que se ha empezado y un menú en el cual se encuentran los módulos que conforman este sistema.

Fig. 2.4.- Pantalla principal del sistema.

Para tener un registro de que usuario se conecta al sistema, desde que máquina lo hace y a que hora se conecto, se utiliza una base de datos llamada proyecto, en la cual a su vez están alojadas dos tablas que son de gran ayuda para el buen funcionamiento del sistema.

Una de ellas es la tabla bitácora, en la cual se almacenan los datos del usuario que se conecto, así como la hora y desde que máquina lo hizo, esto ayuda para tener un registro de las conexiones que se tienen en nuestro sistema y en caso de alguna anomalía poder deslindar responsabilidades.

La otra tabla es la de usuario, en esta se guarda el login, el password, los privilegios de cada usuario, y la vigencia de esta cuenta.

2.4.2 Administración de Usuarios.

En este módulo la administración de usuarios consta de 4 diferentes apartados que podemos encontrar en la administración de usuarios, como lo son: listar usuarios, crear usuarios, borrar usuarios y modificar usuarios, la fig. 2.5 describe de una manera visual como se encuentra la pantalla de administración de usuarios.

Este módulo sólo puede ser utilizado por aquel usuario que tiene el privilegio de crear otros usuarios, de lo contrario sólo podrá ser capaz de listar usuarios ya que este apartado esta disponible para cualquier usuario válido en el sistema.

Fig. 2.5.- Apartado de administración de usuarios.

Listar usuarios.

Este apartado al darle clic en la liga correspondiente muestra sólo un listado de todos los usuarios que se tienen en el sistema, así como los privilegios con los que cuenta, además de mostrar su fecha de vigencia de la cuenta. La fig. 2.6 da un panorama más exacto de cómo muestra el listado de usuarios el sistema.

Fig. 2.6.- Lista de usuarios en el sistema.

Creación de usuarios.

En este apartado es en donde se crean los usuarios del sistema, los cuales serán asignados a las personas para que estas utilicen los privilegios de estos usuarios y realicen su trabajo.

En esta página pedirá los siguientes datos para realizar la creación de usuarios correctamente:

- *Nombre:* Aquí se pone el nombre del usuario, aunque se debe tomar en cuenta algunas restricciones para que el nombre sea válido y el sistema permita crearlo:

  - El nombre del usuario no debe existir, es decir, que no pueden existir dos usuarios con el mismo nombre.
  - La longitud del nombre del usuario debe estar entre 3 y 15 caracteres.
  - No debe contar con caracteres extraños[56] los cuales no son validos en el nombre.

- *Password:* Se pone el password que se le asigna al usuario que se quiere crear, pero también cuenta con la restricción de que su longitud debe ser de 8 a 15 caracteres.

- *Crear BD:* Este es un privilegio que se le puede asignar al usuario que se esta por crear, por default no tiene este privilegio, pero si se marca esta casilla automáticamente tendrá el privilegio de crear bases de datos.

- *Crear usuarios:* Es otro privilegio que se le puede asignar al usuario y este lo que le permite al usuario es que pueda ser capaz de crear otros usuarios.

- *Grupo:* Aparece una caja desplegable donde aparecen el nombre de los grupos existentes, si no existe ningún grupo aparecerá por default la palabra sin grupo.

[56] Los caracteres extraños se refiere a los caracteres que no son validos en el sistema y por lo tanto no se pueden utilizar.

- *Vigencia:* Esta es la fecha en la cual el usuario dejara de existir o de poder conectarse al sistema, la fecha debe ser válida para que se pueda asignar la vigencia, es decir, por default da la fecha en la que se esta creando el usuario, y si la fecha es menor o es una fecha incorrecta entonces no permitirá la creación del usuario.

En la fig. 2.7 resume todo lo anterior.

Fig. 2.7.- Pantalla de creación de usuarios.

Una vez que ya se proporcionaron dichos datos y que no se ha roto ninguna restricción antes mencionada entonces mostrara una pantalla como se observa en la fig. 2.8, de lo contrario marcara un error y no permitirá la creación de dicho usuario.

Fig. 2.8.- Pantalla de usuario creado.

Borrado de usuarios.

En este apartado se muestra en forma de lista los usuarios que pueden ser borrados como se muestra en la fig. 2.9, y para que pueda ser borrado alguno de ellos se deberá escoger un usuario y dar clic en el botón que dice borrar.

Fig. 2.9.- Pantalla de usuarios posibles a borrar.

Si no se escoge ningún usuario y se da clic en el botón de borrar enviara un aviso que dirá que se debe escoger un usuario para poder borrarlo y si se escoge un usuario el mensaje mostrado será que el usuario fue borrado. Aquí existe una restricción más para que un usuario pueda ser borrado, la cual es que el usuario que se quiere borrar no deberá ser el dueño de ninguna base de datos, ya que si es así no se podrá borrar dicho usuario hasta que se borre la base de datos o se cambie de dueño a la misma. Por lo tanto aparecerá una pantalla como se observa en la fig. 2.10, la cual indica que el usuario no puede ser borrado debido a que es dueño ya sea de una o más bases de datos y dará un listado de esas bases de datos.

Fig. 2.10.- Pantalla de error al tratar de borrar un usuario.

Modificación de usuarios.

Al igual que en borrado de usuarios aparece un listado de los usuarios que se pueden modificar y una vez que se escogió el usuario a modificar se da clic en el botón que dice modificar, esto se muestra en la fig. 2.11.

Fig. 2.11.- Pantalla usuarios posibles a modificar.

Después de escoger al usuario que se va a modificar aparece una pantalla como la que se observa en la fig. 2.12, en la cual se muestra el nombre del usuario, el cual no puede ser cambiado y los privilegios con los que cuenta dicho usuario, y es sobre de estos privilegios sobre los que se pueden hacer modificaciones, sólo se tiene que recordar que cumplan con las restricciones de cada uno de los campos, como por ejemplo en la vigencia, que no debe ser una fecha anterior a la actual y que debe ser una fecha válida.

Fig. 2.12.- Pantalla de modificación de usuarios.

Una vez habiendo hecho las modificaciones pertinentes y cuidando que no se infrinja ninguna restricción, entonces se da clic en el botón que dice enviar para que estos cambios surjan efecto en el sistema y nos aparezca una pantalla como la que aparece en la fig. 2.13, mostrando el nombre del usuario y sus nuevas modificaciones en cuanto a privilegios se refiere.

Fig. 2.13.- Pantalla de confirmación de modificación de usuario.

2.4.3 Administración de Bases de Datos.

Este módulo para poder ser utilizado tiene una restricción la cual es que el usuario que lo quiera utilizar tendrá que ser un usuario que tenga el privilegio de creación de base de datos, de lo contrario si no cuenta con este privilegio aparecerá un mensaje de que no puede utilizar alguno de sus apartados ya que no cuenta con el privilegio necesario. Los apartados en los que se divide este módulo son: listar, crear, modificar y borrar base de datos como se puede observar en la fig. 2.14 donde aparecen los apartados que lo conforman.

Fig. 2.14.- Apartado de administracion de base de datos.

Listar base de datos.

En el listado de base de datos sólo se muestra un listado de las bases de datos que están actualmente en el sistema así como el dueño de cada una de esas bases de datos como se muestra en la fig. 2.15.

Fig. 2.15.- Listado de bases de datos en el sistema.

Crear base de datos.

En este apartado del módulo de administración de bases de datos, es donde se crean las bases de datos, pero se debe tener en cuenta algunas restricciones, por ejemplo en el nombre que se le va a asignar a la base de datos no debe tener caracteres extraños y su longitud debe ser entre 3 y 15 caracteres.

Cuando se crea la base de datos también se le puede asignar un dueño, que por default es el usuario que esta creando la base de datos, pero para poder asignar la base de datos a un usuario diferente al que esta creando la base de datos, el usuario que esta creando la base, además de tener el privilegio para crear base de datos y así poder utilizar este módulo, debe contar con el privilegio de crear usuario ya que sin este privilegio no podrá asignar un usuario diferente a él, esto se puede observar en la fig. 2.16.

Una vez que se da clic en el botón enviar mostrara un mensaje en el que dirá que la base de datos fue creado, pero si existe algún error en el nombre de la base de datos mostrara un mensaje que hará alusión al error que se tiene en el nombre de la base de datos.

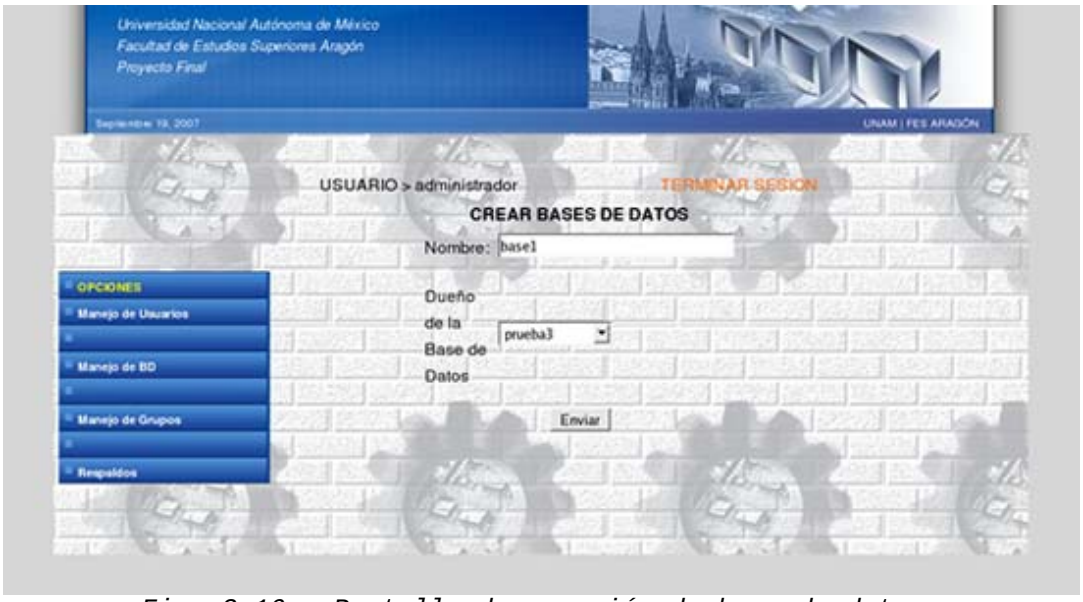


Fig. 2.16.- Pantalla de creación de base de datos.

Modificar base de datos.

En la modificación de base de datos, además de tener el privilegio de creación de base de datos se debe contar con el privilegio de crear usuarios, ya que en la modificación lo único que se va hacer es modificar el dueño de alguna base de datos, en la fig. 2.17 aparece una pantalla donde se muestra un listado de la bases de datos existentes en el sistema las cuales pueden ser modificadas.

Fig. 2.17.- Pantalla de posibles bases de datos a modificar.

Una vez que se escoge la base de datos a modificar se da clic en el botón modificar y envía a otra pantalla en donde aparece el nombre de la base de datos y el dueño de la misma, como se observa en la fig. 2.18, el cual esta en una lista desplegable de manera predeterminada, pero si se quiere cambiar el dueño de la base de datos basta con escoger otro usuario de la lista desplegable para cambiarlo por el actual.

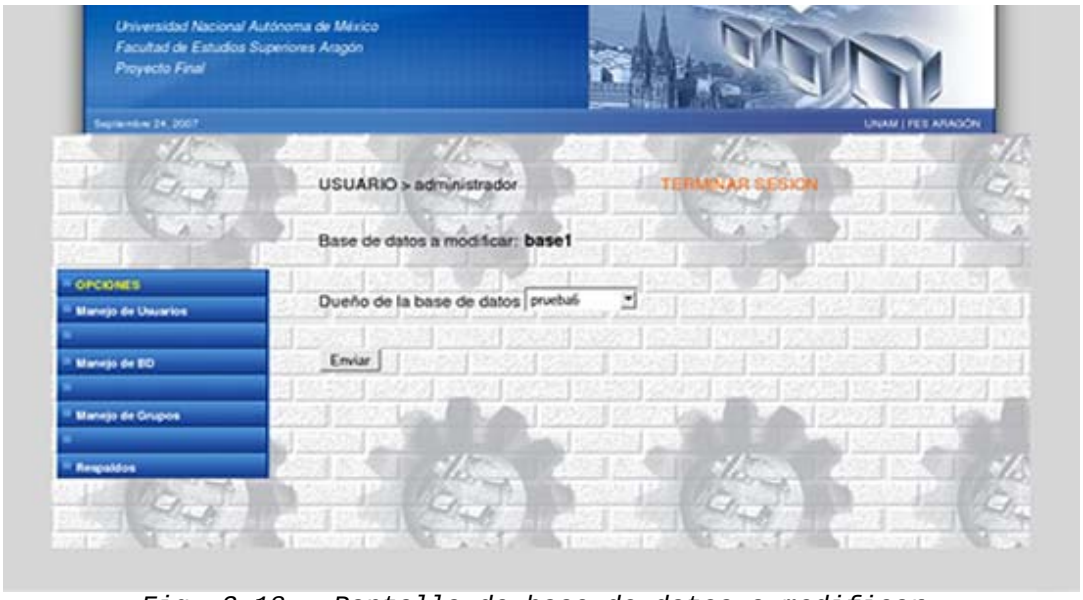


Fig. 2.18.- Pantalla de base de datos a modificar.

Ahora sólo basta con dar clic en el botón enviar para terminar la modificación de dueño de la base de datos, como se muestra en la fig. 2.19, aquí se muestra la base de datos y el nuevo dueño, este apartado es importante porque si se quiere borrar un usuario y es dueño de alguna base de datos no se puede borrar al usuario, y aquí se le cambia el dueño a esa base por un usuario diferente para poder borrar al usuario que ya no se quiere.

Fig. 2.19.- Pantalla de base de datos modificada.

Borrar base de datos.

Para borrar una base de datos se requiere tener el privilegio de creación de base de datos, y en la pantalla aparece un listado de las bases de datos que se pueden borrar en una caja desplegable como se observa en la fig. 2.20, una vez que se haya seleccionado alguna base de datos para borrarla simplemente bastara con dar clic en el botón borrar para que esta acción se realice.

Fig. 2.20.- Pantalla de posibles bases de datos a borrar.

2.4.4 Respaldos.

Los respaldos son muy importantes para el administrador de base de datos ya que le ayudan a tener un resguardo de los datos por si llega ocurrir algún error ya sea en una base de datos o en todo el sistema, y se tiene implementado un buen sistema de respaldos el administrador de base de datos podrá recuperar rápidamente los datos y tendrá una perdida mínima de información.

Este módulo consta de dos apartados como se aprecia en la fig. 2.21 los cuales son el respaldo total y el respaldo de base de datos.

Fig. 2.21.- Apartado de creación de respaldos.

Respaldo total.

Con respaldo total se hace referencia a que se realiza un respaldo general de todas las bases de datos y por ende de todas las tablas y objetos del sistema.

Para poder realizar un respaldo de estos, el usuario que lo quiera realizar tendrá que tener el privilegio de super usuario, ya que para realizar este tipo de actividad se requiere tener privilegios muy amplios.

Para realizar este respaldo basta con dar clic a la liga de respaldo total y así se mandara llamar un programa el cual es un script de shell, el cual permite ejecutar la instrucción de *pg_dumpall*[57] la cual sólo se puede ejecutar desde fuera del manejador de base de datos, es decir, desde el sistema operativo, ya que postgresql no proporciona una instrucción SQL para realizar esta tarea desde dentro del sistema manejador de base de datos.

Una vez que se ha realizado el respaldo aparecerá un mensaje indicando el nombre del archivo del respaldo, así como la ruta del directorio donde se creo como se muestra en la fig. 2.22.

Fig. 2.22.- Pantalla de aviso de respaldo total creado.

Respaldo de base de datos.

A veces no se quiere respaldar todo el sistema, si no más bien respaldar ciertas bases de datos que son criticas para nosotros, así que este apartado permite escoger la base de datos de la cual se quiere realizar un respaldo, la fig. 2.23 muestra la interfaz que le aparece al usuario.

Para poder realizar esta tarea de respaldar una base de datos se debe contar con el privilegio de ser un super usuario para poder respaldar cualquier base de datos, o ser el dueño de esa base de datos para poder respaldarla.

[57] pg_dumpall es un comando que se utiliza para realizar un respaldo total del manejador de la base de datos.

Fig. 2.23.- Pantalla de respaldos posibles de base de datos.

Si se es super usuario se mostrara un listado de todas las bases de datos disponibles en el sistema, y si solamente se es dueño de alguna base de datos se mostrara dicha base de datos para poder respaldarla.

Al igual que en el respaldo total también el sistema se apoya de un script de shell el cual ejecuta el comando *pg_dump*[58] y al cual le pasa como parámetro la base de datos que se quiere respaldar, y una vez que se ejecuta este programa muestra un mensaje informando del nombre del respaldo de la base de datos y su ubicación en donde fue generado como se muestra en la fig. 2.24.

Fig. 2.24.- Pantalla de aviso de creación de respaldo de BD.

[58] pg_dump es un comando para realizar el respaldo de alguna base de datos en particular.

2.4.5 Grupos

Este módulo esta conformado por los apartados de listar, crear y borrar grupos. Este módulo como ya se menciono sirve para crear grupos y ahí poder ingresar a los usuarios que tienen privilegios iguales y así poder otorgar privilegios hacia los objetos como tablas a todo el grupo y esto ayuda a simplificar el estar gestionando los privilegios de los usuarios individualmente.

Listar grupos.

En este apartado simplemente se muestra un listado de los grupos existentes en el sistema, y se mostraría algo parecido a lo que se observa en la fig. 2.25.

Fig. 2.25.- Lista de de grupos en el sistema.

Crear grupos.

Para crear un grupo simplemente basta con tener privilegios de creación de usuarios para poder utilizar este apartado, y poner un nombre al grupo que sea diferente de uno que ya existe o un nombre válido para el grupo el cual no debe contener caracteres especiales además de contar con una longitud de entre 3 y 15 caracteres, la pantalla que aparecerá en el sistema es como la que se muestra en la fig. 2.26.

Si no existe ningún error en el nombre del grupo el grupo se creara, de lo contrario mostrara un mensaje con el error que ha infringido en el nombre del grupo.

Fig. 2.26.-Pantalla de creación de grupo.

Borrar grupos.

Al igual que para crear un grupo, para borrarlo también se debe contar con el privilegio de creación de usuario. Primero el sistema muestra un listado de los grupos existentes y que se pueden borrar, simplemente basta con seleccionar uno de los grupos y dar clic en el botón borrar, ya que si no se selecciona un grupo el sistema informara que se debe seleccionar un grupo para poder borrarlo, pero si se selecciona automáticamente indicara que el grupo fue borrado.

Fig. 2.27.- Lista de posibles grupos a borrar.

Este sistema Web servirá para que el administrador de base de datos que utilice un manejador de base de datos postgresql, implemente este sistema y se ayude en él para que administre a los usuarios, bases de datos, respaldos y grupos de una forma más rápida, eficiente y sin tener que aprenderse los comandos para la creación de estos módulos.

CONCLUSIÓN

La comprensión de todos estos conceptos definidos son de vital importancia para los administradores de bases de datos, para con ellos tomar mejores decisiones y administrar los recursos de una manera más eficiente.

Con la comprensión de estos conceptos y con ayuda de otras herramientas fue posible la creación del sistema Web para la administración de bases de datos, usuarios y respaldos el cual es de gran ayuda para las personas encargadas de administrar estos componentes, y esto puede ayudar a alguna empresa a poder administrar o gestionar de una forma más rápida la información, así con esto se lograra de una forma más fácil la administración sin tener que adentrarnos tan profundamente y sin tener tantos conocimientos en el manejador de base de datos.

Las herramientas que se utilizaron para la construcción de este sistema, se escogieron gracias a su simplicidad, además de que son muy compatibles entre sí, como por ejemplo el servidor Web apache, el lenguaje de programación PHP y la base de datos Postgresql ya que se utilizan mucho gracias a su facilidad de uso y además de que son software libre.

Aquí sería importante apuntar que el SMBD que se debe utilizar para administrar de una manera eficiente nuestra información, es aquel con el que más sepamos trabajar ya que con ello le podremos sacar el mayor jugo a todas sus herramientas con las que cuenta, pero también se debe reflexionar para que se va a utilizar este SMBD ya que no es lo mismo el administrar miles de registro para una tienda, que administrar millones de registros en las empresas.

También es importante decir que la mayoría de las empresas en la actualidad utilizan los sistemas manejadores de bases de datos comerciales ya que la empresa que les provee el servicio aparte de proporcionarles el servicio le da soporte técnico para el sistema manejador de base de datos. Esa es la principal razón por la que se utilizan estos sistemas, aunque los sistemas de código libre como por ejemplo mysql o postgresql están siendo poco a poco aceptados gracias a que estos productos son tan buenos como los comerciales.

BIBLIOGRAFÍA

- Apuntes del diplomado de Administración de Bases de Datos
  DGSCA Centro Coapa.

- PHP 4 a través de ejemplos
  Robles Torres Felipe
  Alfa-Omega

- Como programar en Java 5° Edición
  Deitel M. Harvey and Deitel J. Paul
  Prencite Hall

MESOGRAFÍA

- http://usuarios.lycos.es/cursosgbd/UD3.htm

- http://www3.uji.es/~mmarques/f47/apun/node83.html

- http://mysql.conclase.net/curso/index.php?cap=002