UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

El Ámbito del Administrador de Sistemas

Trabajo escrito en la modalidad de seminarios y cursos de especialización y capacitación profesional que para obtener el titulo de:
INGENIERO MECÁNICO ELECTRICISTA

Presenta:
FAUSTO ALBERTO FERNÁNDEZ MÁRQUEZ

Asesor:
ING. JUAN GASTALDI PEREZ

MEXICO, 2009

Dedicatorias

Los sueños son el motor de la vida. Cuando se tiene un sueño y se dirige un esfuerzo por alcanzarlo existe la posibilidad de que se realice. Mientras exista la posibilidad se debe intentar. El lograr este sueño se lo debo:

A la vida que me regalo nuevamente la oportunidad de lograr este sueño el cual en alguna ocasión perdí la dirección para encontrarlo.

A la universidad que me transmitió los conocimientos y la formación que me permitieron participar con éxito en actividades que no fueron del ámbito de mi profesión.

A la Facultad de Estudios Superiores Aragón por ofrecer nuevas modalidades de titulación que me motivaron para volver al camino de este sueño.

Al personal Administrativo que siempre ha estado disponible para ofrecer sus servicios a los estudiantes para cumplir con los trámites requeridos.

Al personal Docente que con calidad y esmero han transmitido sus conocimientos sin restricción alguna.

A mis familiares y amigos, que en el fondo es lo mismo, que por su alegría, momentos de convivencia y motivación han hecho que los obstáculos para alcanzar este sueño no parecieran insolucionables e inclusive los hicieran agradables.

Y finalmente, dedico la presente a mi esposa. Lucy y a mis hijas, Andrea, Mariana y Ángela, por el sacrificio del tiempo que tenia que dedicar a ellas durante todo este proceso.

Gracias.

Contenido

Introducción

La computación e informática ha evolucionado e incorporándose en todos los ámbitos, desde el social hasta el técnico, en la iniciativa publica y privada tan sencillas como juegos o tan complejas como para administrar grandes grupos financieros, tan aisladas como para un simple usuario o tan distribuidas como aplicaciones a través de Internet.

Esta evolución y amplitud de aplicaciones se debe a la demanda que existe de la informática y a que la tecnología avanza rápidamente a un menor costo.

Siendo tan amplia la influencia de la informática es muy probable que profesionales de otras áreas se involucren en este mercado laboral. En particular a los ingenieros se les presenta mayores oportunidades; tal vez debido a su capacidad analítica, conocimientos de matemáticas, mente racional e ingenio.

Lo extenso del conocimiento y la amplitud de las aplicaciones de la informática hace imposible que se pueda reunir todo el conocimiento global y actualizado en una sola obra. Es por esto que este trabajo no pretende ser una guía de referencia; más bien es un compendio de términos, componentes y actividades más comunes que un administrador de sistemas tiene que realizar.

Se intentó cubrir todas las actividades necesarias para la administración del área de sistemas pero no fue posible. Sin embargo se trataron los temas más importantes.

A pesar de que en esta obra se trata de aplicaciones específicas vigentes en determinada fecha, el conocimiento adquirido será de utilidad porque cualquier aplicación, sin importar cual es o a que versión pertenece, realiza actividades similares. Esto es, adquirimos un conocimiento intuitivo.

A continuación se describe esta obra:

En el Capitulo I empezamos con una breve descripción de los componentes y términos básicos del ambiente de sistemas.

En el capítulo II se trata el sistema operativo. Cualquier computadora sin sistema operativo no es más que un montón de fierros. Se presenta desde la carga del sistema operativo hasta el uso de utilerías prácticas en la administración de este, pasando por la administración de usuarios y mantenimiento básico del sistema.

En el capitulo III se trata de una de las aplicaciones más importantes de la informática: El Administrador de Bases de Datos. Esta aplicación permite administrar y manipular la información para que nos auxilie en la toma de decisiones. Este capítulo contempla la teoría de las bases de datos, las diferentes opciones entre éstas y su administración.

En el capítulo IV tratamos desde la carga de una base de datos. Explotamos la potencia de los Sistemas Administradores de Bases de Datos para ofrecer mayor memoria, procesos más veloces, restricciones de recursos, procesos de auditoria, etc. que algunas aplicaciones demandan. También se presentan algunas opciones por medio de las cuales se pude acceder a los datos que contienen las bases de datos.

Posteriormente, en el capítulo V, se resalta la importancia que tiene la tecnología de la Información para la organización. Las decisiones de tecnología no se toman aisladas en la organización; se deben tomar en coordinación con la Administración de la empresa. Estas decisiones deben seguirse con normas que dictan Institutos Internacionales de Auditoría. Estas normas contemplan desde la adquisición de bienes y servicios de información como la administración que deben de seguir los Sistemas Administradores de Bases de Datos para evitar riesgos de cualquier tipo. En este capitulo también se menciona los riesgos que existen por virus, hackers y otros tipos de accesos no autorizados y qué se debe hacer para reducir los riesgos.

En el capítulo VI, veremos algunas de las herramientas que se usan para modelar y documentar los activos informáticos. Esto es una práctica recomendada que se usa en todas las fases del desarrollo de sistemas. Esta documentación sirve como medio para reducir problemas, costos y mantener informadas a las personas interesadas.

Y finalmente, en el capítulo VII, tratamos de algunas aplicaciones que no serían posibles si no existiera la nueva tecnología que existe en: procesadores, memoría RAM, Sistemas

Operativos, Disco y Administradores de Bases de Datos. Y por último se menciona las Herramientas CASE que son herramientas que nos auxilian en el diseño de las bases de datos

Capítulo I. Conceptos Básicos

Principios de la Informática

La informática es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales. Informática es un vocablo proveniente del francés informatique, acrónimo de las palabras information y automatique. En lo que hoy conocemos como informática confluyen muchas de las técnicas y de las máquinas que el hombre ha desarrollado a lo largo de la historia para apoyar y potenciar sus capacidades de memoria, de pensamiento y de comunicación.

La informática se utiliza en diversidad de tareas, por ejemplo: elaboración de documentos, control de procesos, robots industriales, telecomunicaciones, vigilancia, así como el desarrollo de juegos y multimedios.

En la informática convergen los fundamentos de las ciencias de la computación, la programación y las metodologías para el desarrollo de software, así como determinados temas de comunicaciones y electrónica. Se entiende por informática a la unión sinérgica del cómputo y las comunicaciones.

Definición de Datos

La materia prima o insumo de la informática son los datos que es la representación mínima de información.

Un dato es la unidad mínima de información, hechos sin evaluar o un valor sin significado. Es una representación simbólica (numérica, alfabética, etc.), atributo o característica de una entidad. El dato no tiene valor semántico (sentido) en sí mismo, pero convenientemente tratado (procesado) se puede utilizar en la realización de cálculos o toma de decisiones.

De forma genérica se dice que un dato se puede definir como un hecho aislado y en bruto, que debe ser procesado por varias operaciones para obtener resultados relacionados con la evaluación e identificación de personas, eventos y objetos.

Definición de la Información

La información representa un conjunto de datos relacionados en un contexto determinado que constituyen una estructura de mayor complejidad (por ejemplo, un capítulo de un libro de ciencias) que tienen un significado del cual podemos obtener conocimientos para una futura toma de decisiones.

La información se obtiene asociando los hechos, la adición o el procesamiento de los datos, proporcionan el conocimiento o entendimiento de ciertos factores.

La información 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.

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. La información debe tener las siguientes características:

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

Software

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

- Ø Programas del Sistema: controlan la operación de la computadora. Entre estos se encuentra: Sistemas Operativos, Compiladores e Interprete de comandos
- Ø Programas de Aplicación: Resuelven problemas para los usuarios. Entre estos se encuentran: Editores, Aplicaciones Bancarias, Aplicaciones de oficina, etc.

El Sistema Operativo: es el programa fundamental entre los programas de sistema, controla todos los recursos de la computadora y proporciona la base sobre la que pueden escribirse los programas de aplicación. Es un nivel de software por encima del hardware que controla todas las partes del sistema y presenta al usuario una interfaz o máquina virtual que es más fácil de usar que el hardware subyacente.

Lenguaje de programación

Su función es proporcionar instrucciones al sistema de la computadora para que pueda realizar una actividad de procesamiento. Cada lenguaje de programación utiliza un grupo de símbolos o reglas que tiene un significado específico a eso se le llama sintaxis

Estos lenguajes también han evolucionado en cinco generaciones:

Generación	Descripción
Primera	Los primeros procesadores se programaban directamente en código binario. Cada modelo de procesador tiene su propio código, por esa razón se llama lenguaje de máquina.
Segunda	Los lenguajes simbólicos, asimismo propios de la máquina, simplifican la escritura de las instrucciones y las hacen más legibles. También se les conoce cono ensambladores.
Tercera	Los lenguajes de alto nivel sustituyen las instrucciones simbólicas por códigos independientes de la máquina, parecidas al lenguaje humano o al de las Matemáticas. Como COBOL , FORTRAN o C.
Cuarta	Se ha dado este nombre a ciertas herramientas que permiten construir aplicaciones sencillas combinando piezas prefabricadas
Quinta	Se llama así a veces a los lenguajes de la inteligencia artificial. Estos combinan la creación de códigos basadas en reglas, la administración de reutilización y otros avances. Otra característica es que se le ordena a la computadora realizar un propósito en vez de instruirla para hacerlo.

Conceptos del Hardware

La historia de la computación típicamente se divide en cinco generaciones marcadas por avances significativos en el hardware. Los sistemas operativos han tenido una constante evolución desde sus inicios. Esta gran evolución se ha presentado en un tiempo relativamente corto (desde 1945)

Evolución Computadoras

Primera generación (1945 - 1955)

- Ø Bulbos y conexiones.
- Ø Programación en lenguaje de máquina absoluto o realizando directamente las conexiones eléctricas.
- Ø Alrededor de 1950 se introducen las tarjetas perforadas.

Segunda generación (1955 - 1965)

- Ø Transistores
- Ø Sistemas de procesamiento por lotes.
- Ø FMS(Fortran Monitor System), IBSYS
- Ø Los programas y datos se entregaban en tarjetas, se acumulaban y luego eran procesados todos juntos por la máquina, buscando minimizar los tiempos muertos.

Tercera generación (1965 - 1980)

- Ø Circuitos integrados
- Ø El sistema 360 de IBM unifica computadoras comerciales y científicas en una sola línea de máquinas con software compatible.

- Ø Se introduce la multiprogramación, que divide la memoria en partes y ejecuta un programa distinto en cada una.
- Ø El spooling permite la operación simultánea y en línea de periféricos.
- Ø El tiempo compartido (Timesharing) es una variante de multiprogramación que habilita a cada usuario una Terminal en línea.
- Ø MULTICS (MULTiplexed Information and Computing Service), un gigantesco sistema operativo, fracasa en su construcción pero aporta muchas ideas que hacen surgir UNIX (Desarrollado por Ken Thompson en una PDP-7).

Cuarta generación (1980-1990)
- Ø LSI. Large Scale Integration Circuit
- Ø Estaciones de trabajo y computadoras personales.
- Ø Sistemas operativos DOS y UNIX.
- Ø Software amigable con el usuario.
- Ø Sistemas operativos de red, con varias computadoras interconectadas que pueden ser accedidas por un mismo usuario.
- Ø Sistemas operativos distribuidos, compuestos por varios procesadores que se presentan al usuario como un sistema único.

Quinta Generación (1990-)
- Ø Se desarrollaron las supercomputadoras
- Ø Software de alto nivel (Sistemas expertos, Inteligencia Artificial, etc.)
- Ø Capacidad de comunicarse con la computadora en un lenguaje más cotidiano
- Ø Red más amplia de comunicaciones

Componentes

Son componentes eléctricos, electrónicos, ópticos y mecánicos que permiten la manipulación de datos permitiendo almacenarla, manipularla a gran velocidad y precisión. En la actualidad existen computadoras para todo uso y tamaño y es usado en cualquier ámbito laboral y profesional.

El hardware de la computadora esta formada de varios componentes diferentes, como el CPU, memoria y disco, cada una con un propósito especifico. Para que estos componentes trabajen juntos como un equipo, ellos requieren ser administrados por un sistema operativo.

Memoria RAM

La memoria RAM es la memoria principal de la computadora, también conocida como memoria física. Los programas y datos deben ser cargados dentro de la memoria física para que el sistema las procese. El enunciado “El sistema tiene 64 MB de memoria” se refiere a la cantidad de memoria física o RAM que el sistema tiene instalada.

Es importante la velocidad de acceso en nanosegundos (ns), que es el tiempo que le lleva al CPU tener acceso a la RAM. Entre más nanosegundos tarde más lento es el acceso.

Un programa de software reside en el disco duro y cuando es activado, una imagen o copia del programa es cargado en la memoria RAM. El programa se queda en la RAM el tiempo que sea necesario. Cuando el programa no es requerido por un tiempo, se pueden sobrescribir copias de otros programas. Si el sistema reinicia o experimenta una baja de energía, los datos de la memoria física son borrados.

CPU

La Unidad Central de procesamiento es el chip lógico que ejecuta las instrucciones recibidas de la memoria física, Esas instrucciones son almacenadas en lenguaje binario.

Las características del procesador son la velocidad de procesamiento (la cantidad de instrucciones que ejecuta por segundo), el bus de datos (la capacidad de transmitir información simultáneamente) y la memoria cache (la capacidad de almacenamiento disponible para operaciones internas).

Dispositivos de entrada y salida. Los componentes de entrada y salida permiten intercambiar información del mundo exterior a las computadoras. Dispositivos de entrada como mouse o teclados y de salida como monitor o discos.

Tarjeta Madre. Es la parte más complicada de actualizar por que es donde se ensamblan los demás componentes.

Tipos de Canal (BUS)

PCI, Interconexión de Componentes Periféricos. Trabaja a 64 bits, un ancho de banda de 133 MBPS y se usa en equipos 80486, Pentium, Pentium Pro hasta Pentium III.

AGP Puerto Acelerado de Gráficos (Accelerated Graphics Port). AGP esta basado en PCI, pero es diseñado sobre todo para las exigencias de rendimiento de gráficos 3-D ya que el controlador de gráficos puede tener acceso directamente a la memoria principal. El canal AGP es de 32 bits de ancho y corre a 66 MHZ

Canal (BUS).- De igual relevancia es el tamaño del canal (expresado en bits) ya que un dato se transmite en paralelo, n-bits a la vez. Entre más bits haya en el canal, más rápido pasarán los datos por la RAM. La cantidad de memoria es un factor significativo con respecto a cuántos procesos a la vez puede manejar el servidor y que tan rápido puede procesarlos.

Disco Duro

Es un dispositivo de almacenamiento magnético en el cual los archivos, directorios y las aplicaciones son almacenados. Para un sitio con uno o más usuarios, el disco duro no representa una consideración primaria en el momento del desempeño. Sin embargo, cuando se incrementa tanto el número de usuarios como la demanda de servicios, el tamaño y tipo del disco duro debe de ser tomado en cuenta.

El tipo de disco duro IDE (Integrated Drive Electronics) es la más usada, son buenas para manejar muchos usuarios pero sólo si accedan éstos un archivo a la vez. Por lo que si queremos tener un sitio grande seria recomendable utilizar un disco duro SCSI (Small Computer System Interface). Este tipo de disco tiene una velocidad de transmisión de información más rápida, permite el acceso a múltiples archivos a la vez, además se dirige más rápido a un espacio swap que un IDE. Se puede encontrar los siguientes tipos de SCSI:

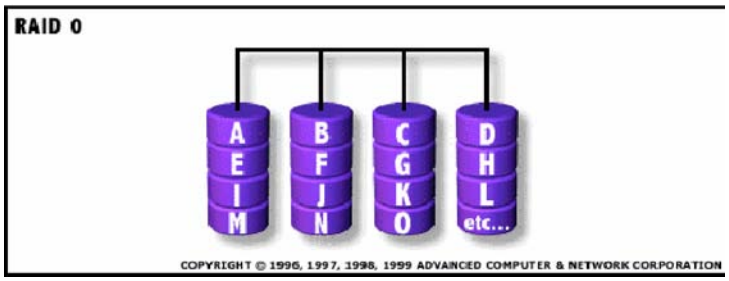
- Ø SCSI-1: Usa BUS de 8 bits, soporta velocidades de transmisión de datos de 4 MBPS.
- Ø SCSI-2: Tiene la misma característica que el SCSI-1, pero usan un conector de 50 terminales en vez de un conector de 25 terminales y soporta dispositivos múltiples.
- Ø Wide SCSI: Usa un cable más amplio (cable de 168 líneas con 68 terminales) para soportar transferencias a 16 bits.
- Ø Fast SCSI: Usa un canal de 8 bits, pero dobla la velocidad de reloj para soportar una velocidad de transferencia de datos a 10 Mbps.
- Ø Fast Wide SCSI: Usa un canal de 16 bits y soporta una velocidad de transferencia de datos de hasta 20 Mbps.
- Ø Ultra SCSI: Usa un canal de 8 bits y soporta velocidades de transferencia de datos de hasta 20 Mbps.
- Ø Wide Ultra SCSI: Usa un canal de 16 bits y soporta velocidades de transferencia de datos de hasta 80 Mbps.

RAID

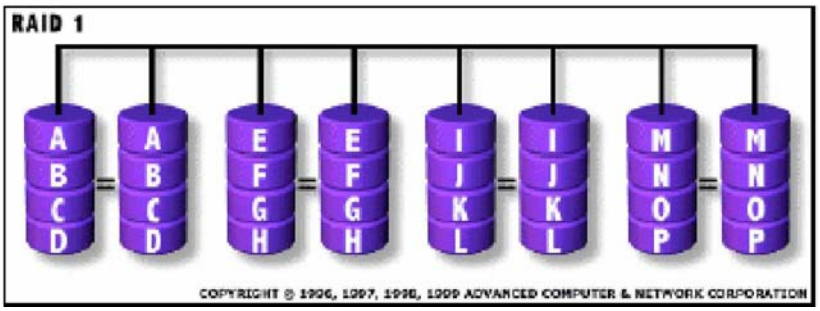
Un RAID contiene varias unidades de disco que funcionan en un arreglo en paralelo para la tolerancia a fallas y el mejoramiento del funcionamiento. Actúa como una enorme unidad de disco, pero almacena la información del sitio en varios discos en lugar de uno. Existen diferentes niveles, los tres más comunes son: 0, 1, 3 y 5:

Nivel 0: Implementa un arreglo de discos fragmentados, los datos son fragmentados en bloques y cada bloque es escrito a una unidad de disco por separado. El funcionamiento de las Entradas/Salidas (I/O) es mejorado al extender la carga de I/O a través de varios canales y discos.

- Ø Ventajas: Es fácil de implementar ya que tiene un diseño sencillo. No gasta tiempo en operaciones sin paridad.
- Ø Desventajas: No es un verdadero RAID ya que no soporta tolerancia a fallas. La falla de un sólo disco puede resultar en la perdida de todos los datos en el arreglo.



Nivel 1: Proporciona un espejo del disco. Para un alto funcionamiento, el control debe ser capaz de llevar a cabo dos lecturas simultáneas o dos escrituras separadas por cada par espejeado. El RAID 1 requiere que un mínimo de 2 discos para implementarlo.



Ventajas

Ø La velocidad de la lectura doble es igual a la de escritura.

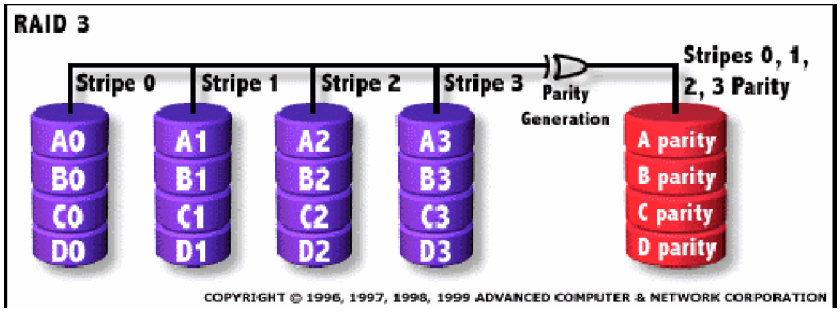
Ø Redundancia del 100 % en los datos, lo que significa que no será necesaria la reconstrucción de los datos en caso de falla, solamente se necesitará una copia del disco de reemplazo.

Ø La velocidad de transferencia por bloques es igual que un disco sencillo. Bajo ciertas circunstancias, el RAID 1 puede soportar varias fallas simultáneas. Es de fácil diseño.

Desventajas

Ø No soporta la remoción de unidades en línea cuando esta implementada por software.

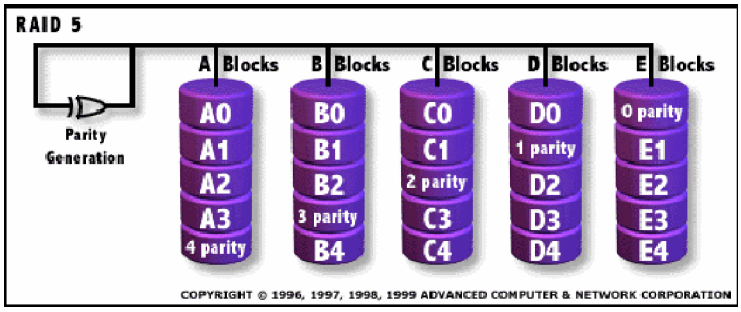
Nivel 3: Igual que el Nivel 0, el bloque de datos es subdividido y escrito sobre los discos de datos. La paridad de partición es generada sobre la Escritura, por lo que se reserva un disco dedicado a la corrección de errores en los datos y revisada en la lectura. Esto proporciona un buen funcionamiento y cierto nivel de tolerancia a fallas. Requiere de al menos tres unidades de discos para su implementación.



Ventajas: Tiene una alta tasa de transferencia de datos en la lectura y escritura. La falla de un disco tiene bajo impacto sobre el rendimiento.

Desventajas: El diseño del controlador es muy complejo. Es difícil de implementar y se necesitaría muchos recursos

Nivel 5: Proporciona datos fragmentados a nivel byte y también corrección errores en los datos por fragmentación. Este resulta excelente para un buen funcionamiento y la tolerancia de fallas. Requiere de al menos tres unidades de discos para su implementación.

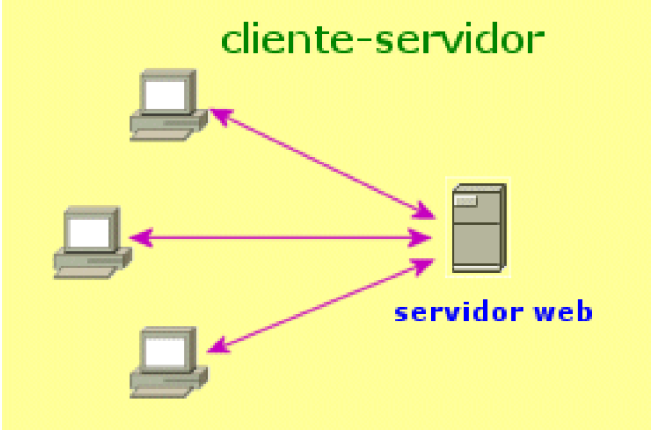


Ventajas: Tiene una alta tasa de transferencia de datos en la lectura y mediana en la escritura.

Desventajas: En caso de falla tiene un mediano impacto en el rendimiento. Es muy complejo de implementar y difícil de reconstruir los datos en caso de falla comparado con el RAID 1. La tasa de transferencia de datos por bloque individual es la misma que la de un simple disco.

Arquitectura Cliente/Servidor

Podemos 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 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.

Los clientes interactúan usualmente de forma gráfica y a su vez se comunican con los procesos auxiliares que se encargan de establecer la conexión con el servidor, de enviar el pedido, de recibir la respuesta, manejar las fallas y realizar actividades tanto de sincronización como de seguridad.

Los servidores proporcionan un servicio al cliente y devuelven los resultados, en algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente.

Una de las cosas por la que se distingue el modelo Cliente/Servidor, es que su procesamiento es distribuido entre aplicaciones independientes.

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). El Servidor procesa las peticiones que hacen los clientes y cuando es posible regresa el resultado. Mantiene la integridad lógica y el acceso de los datos.
- Front-end (Cliente). 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 acceda al servidor de bases de datos se puede generalizar en los siguientes pasos:

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

Capítulo II. Sistemas Operativos

Conceptos de Sistemas Operativos

El sistema operativo interpreta las instrucciones del usuario o de las aplicaciones y dice a la computadora que hacer, es el encargado de proporcionar una asignación ordenada y controlada de los recursos para los programas que compiten por ellos. Manipula las entradas y las salidas, guarda el seguimiento de los datos almacenados en el disco y se comunica con periféricos como el monitor, disco duro, impresoras, modems, etc.

Fundamentos

Los tres principales componentes del sistema operativo Linux son:

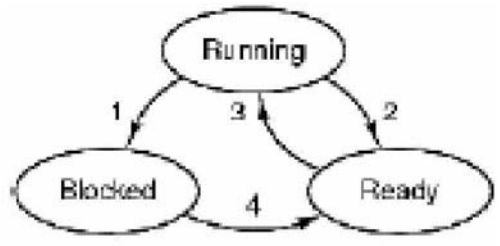
- El Kernel.- El kernel es el núcleo del sistema operativo. Es el programa maestro que administra todos los recursos de la computadora, incluyendo:
  - El sistema de archivos
  - Administración de dispositivos
  - Administración de procesos
  - Administración de la memoria
- El Shell.- El Shell es una interfaz entre el usuario y el kernel. La función primaria del shell es ser un intérprete de instrucciones o comandos. Esto es, el Shell acepta las instrucciones que se ingresan, los interpreta y después las ejecuta. El shell por default en Linux es el BASH (Bourne Again Shell).
- Árbol de Directorios.- El árbol de directorios se refiere al conjunto de directorios que donde se almacena toda la información del sistema operativo.

Procesos

Todo el software ejecutable de la computadora se organiza en varios procesos. Un proceso es tan solo un programa en ejecución, consta del programa ejecutable, sus datos y pila. Los estados en los que un proceso se puede encontrar son:

- En ejecución
- Listo (no existe CPU disponible para el)
- Bloqueado

Las transiciones entre un estado y otro se detallan en la figura.



1) Proceso bloqueado para entrada
2) Scheduler selecciona otro proceso
3) Scheduler selecciona este proceso
4) La entrada es disponible

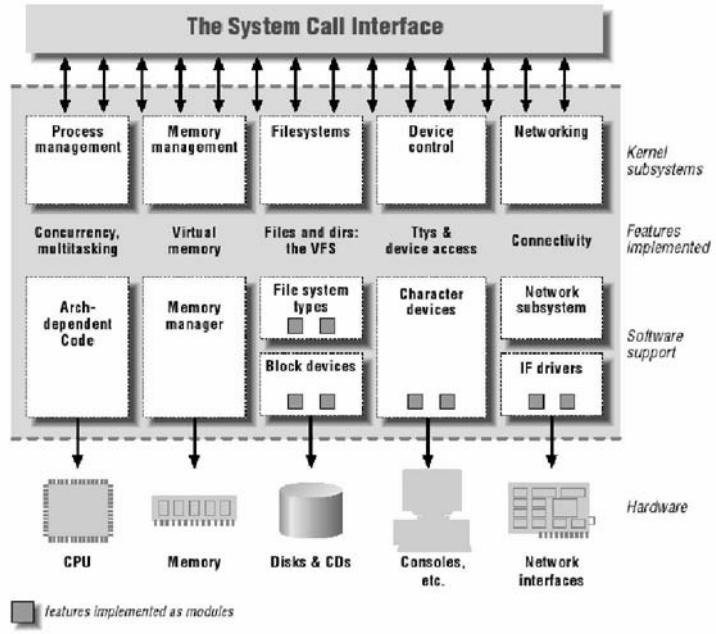
- Comunicación entre procesos.- Los procesos durante su ejecución, rara ves hacen las cosas solos, necesitan comunicarse con otros procesos. Cuando hay problemas de comunicación entre ellos se generan condiciones de competencia.
- Condición de competencia.- Es cuando dos o más procesos leen o escriben en ciertos datos compartidos y el resultado final depende de quién ejecuta qué y en que momento.
- Llamadas al sistema.- Mecanismos de comunicación entre el intérprete de comandos y el sistema operativo
- Archivos.- Secuencia de bits.

Componentes de un Sistema Operativo

Un sistema operativo está formado por varios programas que en conjunto presentan al usuario una vista integrada del sistema, los componentes principales de un sistema operativo son los siguientes módulos:

- Ø Administrador de procesos. (scheduler)
- Ø Administrador de E/S.
- Ø Administrador de la Memoria.
- Ø Manejo del Sistema de Archivos.

La figura muestra los componentes típicos del kernel de GNU/Linux.



Administrador de Procesos.

El planificador de procesos (conocido como scheduler) es el encargado de decidir cual proceso se ejecuta primero y por cuanto tiempo (quantum).

Objetivos

- Ø Equidad: garantizar que el proceso obtiene su proporción justa de CPU.
- Ø Eficacia: mantener ocupado el CPU el 100
- Ø Tiempo de Respuesta: minimizar el tiempo de respuesta para los usuarios interactivos.
- Ø Tiempo de regreso: minimizar el tiempo que deben esperar los usuarios por lotes para obtener resultados.
- Ø Rendimiento: maximizar el número de tareas procesadas por hora.

Tipos de estrategias

Para lograr sus objetivos, el planificador de procesos cuenta con dos tipos de estrategias:

- Ø Planificación apropiativa: permite que procesos ejecutables seas suspendidos en forma temporal.
- Ø Planificación no apropiativa: ejecución hasta terminar (primeros sistemas operativos por lotes)

Algoritmos de planificación de procesos

Cada proceso tiene asignado un intervalo de tiempo de ejecución (quantum). Si el proceso continua en ejecución al final de su quantum, otro proceso se apropia de la CPU. Si el proceso está bloqueado o ha terminado antes de consumir su quantum, se alterna el uso de la CPU. La alternancia entre un proceso y otro necesita cierta cantidad de tiempo para administración (resguardo y carga de registros y mapas de memoria, actualización de varias tablas y listas, etc.). Si el quantum es muy corto, se alternan demasiados procesos, lo que reduce la eficacia de la CPU; pero si es muy largo puede causar una respuesta lenta a las solicitudes interactivas breves.

- Ø Por prioridad hay dos clases de prioridades: estática o dinámica. Si las prioridades no se ajustan de vez en cuando, las clases de prioridad mínima podrían morir de inanición eterna.
- Ø Primero el trabajo más corto.

Opciones de Sistemas Operativos

Existe una gran variedad de sistemas operativo que van desde los propietarios (pertenecientes a algún fabricante en especial), comercial (disponibles en el mercado) y libres (gratis). Entre estas opciones se encuentran:

Unix

Unix es un sistema operativo portable, multitarea y multiusuario; desarrollado en principio por los laboratorios Bell de AT&T, entre los desarrolladores que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy. El inicio del desarrollo de este sistema operativo (uno de los más influyentes en la historia de la computación) fue muy particular, pues nadie habría predicho el éxito de UNIX después de su primera encarnación.

En 1968, el Instituto Tecnológico de Massachusetts, los Laboratorios Bell de AT&T y General Electric trabajaban en un sistema operativo experimental llamado MULTICS (Multiplexed Information and Computing Service), desarrollado para ejecutarse en una computadora central (mainframe) modelo GE-645. El objetivo del proyecto era desarrollar un sistema operativo interactivo que contase con muchas innovaciones, entre ellas mejoras en las políticas de seguridad, la multitarea, la gestión de archivos y la interacción con el usuario. Las primeras versiones contaban con un pobre rendimiento. Los laboratorios Bell de AT&T decidieron desvincularse y dedicar sus recursos a otros proyectos.

Ken Thompson, siguió trabajando para la computadora GE-635 escribió nuevamente el programa, con ayuda de Dennis Ritchie, en lenguaje ensamblador, para que se ejecutase en un ordenador DEC PDP-7. Thompson y Ritchie lideraron un grupo de programadores, entre ellos a Rudd Canaday, en los laboratorios Bell, para desarrollar el sistema de ficheros y el sistema operativo multitarea en sí. A lo anterior, agregaron un intérprete de comandos y un pequeño conjunto de programas. El proyecto fue bautizado UNICS, como acrónimo Uniplexed Information and Computing System, pues sólo prestaba servicios a dos usuarios (de acuerdo a Andrew Tanenbaum, era sólo a un usuario). La autoría de esta sigla se le atribuye a Brian Kernighan ya que era una mutación de MULTICS. Dada la popularidad que tuvo un juego de palabras que consideraba a UNICS un sistema MULTICS castrado (pues eunuchs, en inglés, es un homófono de UNICS), se cambió el nombre a UNIX, dando origen al legado que llega hasta nuestros días.

Hasta ese instante, no había existido apoyo económico por parte de los laboratorios Bell, pero eso cambió cuando el Grupo de Investigación en Ciencias de la Computación decidió utilizar UNIX en una máquina superior a la PDP-7. Thompson y Ritchie lograron cumplir con la solicitud de agregar herramientas que permitieran el procesamiento de textos a UNIX en una máquina PDP-11/20 y como consecuencia de ello consiguieron el apoyo económico de los laboratorios Bell. Fue así como por vez primera, en 1970, se habla oficialmente del sistema operativo UNIX ejecutado en una PDP-11/20. Tanto el sistema operativo como los programas fueron escritos en el lenguaje ensamblador de la PDP-11/20. El 3 de noviembre de 1971 Thomson y Ritchie publicaron un manual de programación de UNIX (título original en inglés: "UNIX Programmer's Manual").

En 1972 se tomó la decisión de escribir nuevamente UNIX, pero esta vez en el lenguaje de programación C. Este cambio significaba que UNIX podría ser fácilmente modificado para funcionar en otros ordenadores (de esta manera, se volvía portable) y así otras variaciones podían ser desarrolladas por otros programadores. Ahora, el código era más conciso y compacto, lo que se tradujo en un aumento en la velocidad de desarrollo de UNIX. Gracias a esto la popularidad de UNIX creció y permitió asentar la filosofía UNIX. AT&T puso a UNIX a disposición de universidades y compañías, también al gobierno de los Estados Unidos, a través de licencias. Una de estas licencias fue otorgada al Departamento de Computación de la Universidad de California, con sede en Berkeley.

La Universidad de California en Berkeley comenzó sus desarrollos en el campo UNIX, añadiendo nuevas características y haciendo modificaciones. Así, en 1975 Ken Thompson promovió el desarrollo y sacó a la luz su propia versión de UNIX, conocida como BSD. Desde entonces BSD pasó a convertirse en la gran competidora de los laboratorios Bell.

Mientras tanto, AT&T creó una división comercial denominada Unix Systems Laboratories para la explotación comercial del sistema operativo. El desarrollo prosiguió, con la entrega de las versiones 4, 5 y 6 en el transcurso de 1975. Ya en 1978, cerca de 600 o más máquinas estaban ejecutándose con alguna de las distintas encarnaciones de UNIX.

La versión 7, la última versión del UNIX original con amplia distribución, entró en circulación en 1979. Las versiones 8, 9 y 10 se desarrollaron durante la década de 1980, pero su circulación se limitó a unas cuantas universidades, a pesar de que se publicaron los informes que describían el nuevo trabajo. Los resultados de esta investigación sirvieron de base para la creación de Plan 9, un nuevo sistema operativo portable y distribuido, diseñado para ser el sucesor de UNIX en investigación por los Laboratorios Bell.

Familias

Como se puede deducir de esta breve reseña histórica, existen varias familias del sistema operativo UNIX que han evolucionado de manera independiente a lo largo de los años. Cada familia se distingue no tanto por sus diferencias técnicas como por sus diferencias en propiedad intelectual. Se observa que todas las familias se han visto contaminadas, directa o indirectamente, por otras familias.

Nombre	Propietario	Características Relevantes
AIX	IBM	esta familia surge por el licenciamiento de UNIX System III a IBM
A/UX	Apple	
AT&T		familia que tuvo su origen en el UNIX de AT&T. Considerada la familia UNIX "pura"
BSD	Público	familia originada por el licenciamiento de UNIX a Berkely
DEC OSF/1	Digital Research	Basado en BSD e intento de standard Open Software Foundation. Actualmente DEC abandonó el OSF pues no resultó un standard y comercializa Digital UNIX (DU). Arquitectura Alpha
IRIX	Silicon Graphics	Similar a AT&T. Arquitectura propietaria
HP-UX	Hewlett-Packard	Híbrido AT&T y BSD, con particularidades propias. Arquitectura propietaria
SCO	The Santa Cruz Operation	Basado en AT&T pero con muchos agregados. Arquitectura Intel x86.
SunOS Solaris	Sun Microsystems	Basado en AT&T, con muchas extensiones. Arquitectura Sparc y x86.
Tru64		actualmente de Hewlett-Packard (antes de Compaq y originalmente de Digital Equipment Corporation)
UnixWare y SCO OpenServer	anteriormente de Santa Cruz Operation y ahora de SCO Group	
UX/4800	NEC	
Xenix		familia derivada de la adquisición de los derechos originales de AT&T primero por parte de Microsoft y de esta los vendió a SCO

Características del UNIX (Y Sus Clones)

- Ø Interactivo: el usuario puede trabajar de forma interactiva, sin tener que esperar un gran tiempo de respuesta
- Ø Multiusuario: varios usuarios pueden trabajar a la vez desde distintos terminales (tiempo compartido)
- Ø Multitarea: más de una tarea a la vez, en diferentes sesiones, modo background
- Ø Multiprocesador: permite más de un procesador
- Ø De propósito general: no es específico para un tipo de trabajos sino que admite todo tipo de aplicaciones
- Ø Está diseñado para crear un entorno de programación sencillo, eficiente y flexible para programadores y diseñadores. Soporta lenguajes de programación como: C, PASCAL, ENSAMBLADOR, COBOL, INFORMIX, ORACLE.
- Ø Unix fue concebido para entornos grandes, potentes servidores de Internet y básicamente, para el mundo empresarial y como ya te estarás imaginando, todo lo anterior hace que un sistema Unix sea demasiado caro para el usuario final.

Linux

Linux, FreeBSD, OpenBSD son clones de Unix, respetan sus normas y sus estándares (POSIX, BSD), pero además gozan de una característica importante son Fuente Abierta y están bajo la cobertura de la GPL, la Licencia Publica General GNU. Esto quiere decir que además de tener la potencia que tienen, son gratis, no pertenecen a ninguna empresa y permiten obtener todo el código fuente.

En 1983, Richard Stallman anunció el Proyecto GNU, un ambicioso esfuerzo para crear un sistema similar a Unix, que pudiese ser distribuido libremente. El software desarrollado por este proyecto, también han sido parte fundamental de otros sistemas UNIX. En 1991, cuando Linus Torvalds empezó a proponer el kernel Linux y a reunir colaboradores, las herramientas GNU eran la elección perfecta. Al combinarse ambos elementos, conformaron la base del sistema operativo (basado en POSIX) que hoy conocemos como GNU/Linux o simplemente Linux. Las distribuciones basadas en el kernel, el software GNU y otros agregados entre las que podemos mencionar a Red Hat Linux y Debian GNU/Linux se han hecho populares tanto entre los aficionados a la computación como en el mundo empresarial. Obsérvese que GNU/Linux tiene un origen independiente, por lo que se considera un 'clónico' de UNIX y no un UNIX en el sentido histórico. GNU/Linux incorpora propiedad intelectual de BSD, gracias a que éste también se libera con una licencia de código abierto denominada Open-source BSD.

La historia de Linux empieza en Finlandia, en el 1991, cuando al Linus B. Torvalds, estudiante de la Universidad de Helsinki se le ocurrió comprarse una PC con procesador 386. Después de observar con el MS/DOS no aprovechaba los recursos de la máquina, decidió usar otro sistema operativo de entonces: Minix. Minix era un pequeño sistema Unix.

Sin embargo, debido a las limitaciones del Minix, Linus decidió reescribir algunas partes del sistema, añadiéndole mayor funcionalidad. Posteriormente, Linus decidió difundir el código fuente por Internet, de manera gratuita y con el nombre de Linux (contracción de Linus y Unix). ¡Linux había nacido! La primera difusión de Linux tuvo lugar el mes de agosto de 1991. Se trataba de la versión 0.01 y por el momento, funcionaba bajo Minix.

Este es el origen de los que conocemos ahora como Linux. Cuando tuvo una versión suficientemente estable comenzó a distribuirla bajo la licencia GPL y solicitó ayuda para hacer pruebas y mejorarlo. Desde entonces Linux ha evolucionado enormemente. El número de ordenadores que funcionan bajo Linux ha aumentado espectacularmente en los últimos años.

El éxito de Linux se debe fundamentalmente a su distribución por Internet que ha permitido la incorporación de los desarrollos de gente repartidos por todo el mundo. Actualmente Linux cuenta con los principales gestores de ventanas, utilidades para Internet, compiladores, editores.

Linux se utiliza con éxito como servidor en muchas empresas y universidades de todo el mundo y cada vez son más los usuarios particulares que se deciden por este. La primera versión estable de Linux fue la 1.0 y apareció en marzo del 1994. Una de las razones de su demanda es porque ofrece herramientas potentes de uso en la red que ningún otro sistema operativo posee.

Inicialmente Linux se diseñó como un clónico de Unix, distribuido libremente para funcionar en máquinas PC con procesadores 386, 486 o sea, para arquitectura x86. En la actualidad funciona sobre otras muchas plataformas como los procesadores Alpha, Sparc, Amiga, Atari, las máquinas tipo MIPS y sobre los PowerPC

Hay que resaltar también que Linux respeta las especificaciones POSIX, pero posee también ciertas extensiones de las versiones System V y BSD de Unix. Esto simplifica notablemente la adaptación de programas desarrollados inicialmente para otros sistemas Unix.

El termino POSIX significa Portable Operating System Interface. Son unas normas definidas por el IEEE y estandarizados por el ANSI y el ISO. POSIX permite tener un código fuente transportable.

Hay que recordar también que el término Linux se refiere al Núcleo del sistema (lo que interactúa con el Hardware de nuestra máquina). Cuando hablamos de todo el conjunto que forman el núcleo y todos los demás proyectos GNU (las shells, compiladores, escritorios y las distintas aplicaciones en general), estaremos hablando ya del Sistema Operativo GNU/Linux.

La clave de la versión indica la fase de desarrollo en que se encuentra:

- Ø Fase de desarrollo: el núcleo cuya estabilidad no está asegurada, es el momento donde se añade funcionalidad al núcleo, optimizaciones y demás. En definitiva, es la fase en la que se desarrolla más el núcleo y se caracteriza por su nombre de versión impar: 1.1 , 1.3
- Ø Fase de estabilización: se trata de coger el núcleo desarrollado en la fase anterior y hacer que este sea lo más estable posible. Aquí las modificaciones son mínimas, se trata más de retoques y pequeños ajustes. Los núcleos estables tiene número de versión par: 1.0, 1.2, 2.0, 2.4 ...

Versión	Año	Usuarios Estimados	Tamaño del kernel (KB)	Hitos
0.01	1991	100	63	Linus Torvalds escribe el kernel de Linux.
0.99	1992	1000	431	Software GNU es integrado al kernel dando como resultado un sistema totalmente funcional.
0.99	1993	20000	938	Dada la cantidad de contribuciones al código, Linus se ve obligado a delegar la responsabilidad de revisión del kernel.
1.0	1994	100000	1017	Se libera la primera versión estable.
1.2	1995	500000	1850	Compatibilidad con procesadores no intel.
2.0	1996	500000	4718	Linux soporta múltiples procesadores, IP masquerading y Java.
2.2	1999	7,500000	10593	El crecimiento de Linux excede a Windows NT.
2.4	2001	10000000	19,789	Linux invade las empresas privadas.
2.6	2003	20-50000000	32476	Linux mejora para cumplir con requerimientos de cualquier sistema operativo comercial, como la mejora en el manejo de hilos (threads).

Características

GNU/Linux posee las siguientes características:

- Ø Portable: el mismo sistema operativo corre en un espectro de máquinas que van desde notebooks a supercomputadoras. Es el único sistema operativo con estas características.
- Ø Flexible: se adapta a muchas aplicaciones diferentes.
- Ø Potente: dispone de muchos comandos y servicios ya incorporados.
- Ø Multiusuario: atiende a muchas personas simultáneamente.
- Ø Multitarea: hace muchas cosas a la vez.
- Ø Elegante: sus comandos son breves, coherentes, específicos para cada tarea y muy eficientes.
- Ø Orientado a redes desde el comienzo.
- Ø Dispone de un estándar (POSIX) que debe cumplir todo sistema operativo que pretenda ser Unix, lo que asegura una evolución predecible y compatibilidad con otros Unix.

Objetivos de GNU/Linux

GNU/Linux fue diseñado teniendo en mente los siguientes objetivos:

- Ø Crear un sistema interactivo de tiempo compartido diseñado por programadores y para programadores, destinado a usuarios calificados. que fuera sencillo, elegante, escueto y consistente.
- Ø Que permitiera resolver problemas complejos combinando un número reducido de comandos básicos.

Filosofía del sistema GNU/Linux

Los objetivos con que se creó determinaron una “filosofía” caracterizada por:

- Ø Comandos cortos, simples, específicos y muy eficientes, que “hacen una sola cosa pero la hacen muy bien”.
- Ø Entrada y salida estandarizadas que permiten la interconexión de comandos. Esto se llama entubamiento (“pipelining”): la salida de un comando es tomada por el siguiente como entrada.
- Ø Todo es un archivo.

Distribuciones LINUX

Nombre	Características Relevantes
CentOS	Es un clon a nivel binario de la distribución Red Hat
Debian	Es una distribución bastante popular que no está desarrollada por ninguna compañía comercial sino que es fruto del trabajo de diversos voluntarios en toda la comunidad de Internet. Es, por lo tanto, una distribución completamente gratis

Fedora Core	El laboratorio de pruebas para la versión comercial de RedHat. Esta distribución es libre
Knoppix	
Mandriva	principiantes o usuarios medios
Red Hat	excelente auto detección de dispositivos, instalador gráfico y un excelente conjunto de aplicaciones comerciales en su distribución oficial
Slackware	Una de las primeras distribuciones. Ha perdido terreno a favor de distribuciones más modernas, siendo relegada a aplicaciones especializadas
SuSE	Esta distribución es la más popular en Europa
Ubuntu	

Instalación de un Sistema Operativo

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, empresa desarrolladora de RHEL ftp://ftp.redhat.com/pub/redhat/linux/enterprise/4/en/os/i386/SRPMS. Alternativa libre a distribuciones comerciales de compañías como Red Hat, Suse y Mandriva.

Instalación

Antes de iniciar una instalación de cualquier sistema operativo, tenemos que revisar que el hardware es compatible con el sistema a instalar y además cumple con ciertos requerimientos técnicos. Para el caso de RHEL o CentOS, esta disponible en la siguiente URL: http://www.redhat.com/hcl En la que encontraremos el hardware compatible con RHEL.

Después de ejecutar el disco de instalación y que éste valide que los requisitos de hardware están cubiertos procede a generar un disco RAM (simulación de disco físico). Lo anterior lo realiza para no dañar o perder información del disco físico hasta que sea validada la configuración.

Dentro del proceso de instalación se debe ejecutar el programa ANACONDA que es el programa que se encarga de validar la integridad de los discos de instalación, iniciar y probar el hardware y de recolectar la información necesaria para la instalación. Al terminar la validación advierte sobre el riesgo de perder información del disco duro.

Unidad Central de Proceso

CentOS no soporta procesadores Intel i386 o inferiores. Estos procesadores son miembros de la familia conocida como x86. CentOS también soporta procesadores que no sean intel (non-Intel como AMD's AMD64, IBM's PowerPC y algunos procesadores usados en mainframes IBM. También soporta los nuevos procesadores Intel Itanium.

Para poder usar CentOS en el escritorio la velocidad mínima del procesador debe ser de 400 MHz con un procesador Pentium II. En modo texto se requiere al menos un procesador de 200MHz. Para cada tipo de procesador existe un set o conjunto de cd's distintos.

Para un desempeño óptimo se necesita por lo menos 256 MB de RAM. Esta es la recomendación de RedHat, más sin embargo la mayoría de la gente instala GNU/Linux en sistemas que van desde 192 hasta 256 MB de RAM. Se puede instalar con 64 MB de RAM pero sin ambiente gráfico (no GUI).

Tipo de Instalación. Los tipos de instalaciones disponibles son:

- Escritorio personal. Ideal para computadoras caseras o laptop's, se instala un entorno de escritorio gráfico con toda la paquetería necesaria para jugar, hacer documentos, hojas de cálculo, presentaciones, editar imágenes.
- Estación de trabajo. Ideal para programadores ya que se instalan las herramientas necesarias para el desarrollo de software y administración del sistema.
- Servidor. Si se necesita un servidor web, de archivos, de impresión, de base de datos o de infraestructura (DHCP, DNS), esta opción es la ideal. Con esta opción no es instala en ambiente gráfico.
- Personalizada. Si deseamos seleccionar paquete por paquete esta opción es la indicada.

Disco Duro

La cantidad de espacio necesaria para la instalación varía de distribución a distribución de acuerdo a la siguiente tabla:

Tipo de Instalación	Espacio mínimo requerido
Personal Desktop	2.3 Gb
Workstation	3.0 Gb
Server	1.1 - 1.5 Gb (sín ambiente gráfico) 6.9 gigabytes (todo incluido)
Custom	0.62 - 6.9 Gb

CentOS como cualquier otro GNU/Linux necesita por lo menos dos particiones para trabajar, / (root) y swap. Aunque las recomendadas son:

Partición	Tamaño
/boot	100 a 150 megabytes
/(root)	500 a 2048 megabytes
(swap)	1.5 veces la cantidad de memoria RAM en sistemas con un mínimo de 500 megabytes
/home	Tan grande como sea necesario; depende del número de usuarios y el tipo de trabajo que realicen
/tmp	Mínimo de 500 megabytes
/usr	Mínimo de 1.7 - 4048 gigabytes, dependiendo de la versión de CentOS y la cantidad de paquetes a instalar
/var	Mínimo de 500 a 2048 megabytes

Acerca de las Consolas Virtuales

El programa de instalación de CentOS ofrece terminales adicionales a las ventanas de diálogo del proceso de instalación. Además de darle la posibilidad de insertar comandos desde el intérprete de comandos del shell, le muestra muchos mensajes de diagnóstico. El programa de instalación despliega estos mensajes en cinco consolas virtuales, entre las que puede cambiarse usando la siguiente combinación de teclas.

Consola	Combinación de teclas	Contenido
1	Ctrl - Alt - F1	diálogo de instalación
2	Ctrl - Alt - F2	intérprete de comandos
3	Ctrl - Alt - F3	registro de instalación (mensajes del programa de instalación)
4	Ctrl - Alt - F4	mensajes del sistema
5	Ctrl - Alt - F5	otros mensajes
7	Ctrl - Alt - F7	pantalla gráfica de X

Los mensajes mostrados en las consolas virtuales pueden ayudarle en el caso de que encuentre problemas durante la fase de instalación de CentOS.

Medios de Instalación

Para arrancar el programa de instalación, se puede usar cualquiera de los siguientes medios (en función del medio compatible con nuestro sistema):

- Ø CD - ROM. Si nuestro equipo soporta una unidad de CD - ROM de arranque y contamos con el conjunto de CD - ROMs de CentOS completo.
- Ø Módulo de memoria USB. Si nuestro equipo soporta el arranque desde un dispositivo USB.
- Ø Disco duro. Si descargamos los discos de la red y los copiamos a nuestro disco duro, los podemos utilizar para la instalación.
- Ø NFS. Si estamos realizando la instalación desde un servidor NFS utilizando imágenes ISO o una copia, podemos utilizar este método. Aun así necesitamos un CD - ROM de arranque y usar la opción de arranque linux askmethod.
- Ø FTP. Si estamos realizando la instalación directamente desde un servidor FTP, utilizamos este método. Aún así necesitamos un CD - ROM de arranque y usar la opción de arranque linux askmethod.
- Ø HTTP. Si estamos realizando la instalación directamente desde un servidor Web HTTP, utilizamos este método. Aun así necesitamos un CD - ROM de arranque y usar la opción de arranque linux askmethod.

En caso que hayamos elegido la opción de instalación vía los set's de cd's, el primer paso es verificar la integridad de los cd's. Esto es únicamente necesario cuando nos queremos asegurar que nuestros discos funcionan sin problema. Este proceso puede tardar hasta 40 minutos. Para elegir una opción apretamos tabulador y luego enter.

El instalar anaconda prueba la tarjeta de video y después de reconocerla y comprobar que el ambiente gráfico puede iniciar sin problemas, la pantalla de bienvenida es desplegada.

A continuación se selecciona el Idioma, el teclado (Para cambiar la configuración del teclado después de haber completado la instalación, se usa el comando SYSTEM-CONFIG-KEYBOARD)

Tipos de sistemas de archivos

CentOS nos permite crear diferentes tipos de particiones las cuales se describen a continuación:

- ext2. Un sistema de archivos ext2 soporta tipos de archivo estándar Unix (archivos regulares, directorios, enlaces simbólicos, etc.). Proporciona la habilidad de asignar nombres de archivos largos, hasta 255 caracteres.
- ext3. El sistema de archivos ext3 está basado en el sistema de archivos ext2 y tiene una ventaja principal, el journaling. El uso de un sistema de archivos journaling reduce el tiempo de recuperación tras una caída ya que no es necesario hacer FSCK al sistema de archivos. El sistema de archivos ext3 está seleccionado por defecto y su uso es bien recomendado.
- volumen físico (LVM). Mediante la creación de una o más particiones LVM físicas nos permite crear un volumen lógico LVM. LVM puede mejorar el rendimiento cuando se utilizan discos físicos.
- software RAID. La creación de dos o más particiones de software RAID nos permite crear un dispositivo RAID (Redundant Array of Independent Disks).
- Swap. Las particiones swap se usan para soportar la memoria virtual. En otras palabras, los datos se escriben en esta partición cuando no hay suficiente RAM para guardar los datos que el sistema está procesando.
- Vfat. El sistema de archivos VFAT Linux es compatible con los nombres largos de archivos de Microsoft Windows en el sistema de archivos FAT.

Particionando el Disco Duro

El particionamiento permite dividir el disco duro en secciones aisladas, donde cada sección se comporta como un propio disco duro. El particionamiento es especialmente útil si ejecuta más de un sistema operativo. Se puede elegir entre realizar un particionamiento automático o un particionamiento manual con DISK DRUID.

Particionamiento automático. El particionamiento automático es útil si se cuenta con espacio libre en nuestro disco duro. Tiene las siguientes opciones:

- Eliminar todas las particiones Linux del sistema. Esta opción elimina sólo las particiones Linux (particiones creadas en una instalación Linux previa), no borrará el resto de particiones del disco(s) duro(s) (tal como VFAT o particiones FAT32).
- Eliminar todas las particiones del sistema. Esta opción elimina todas las particiones del disco duro (esto incluye las particiones creadas por otros sistemas operativos tales como particiones Windows VFAT o NTFS).
- Mantener todas las particiones y usar el espacio libre existente. Esta opción conserva los datos y las particiones actuales, asumiendo que se tiene suficiente espacio disponible en los discos duros.

Particionamiento manual

Si elegimos el particionamiento manual, se debe indicar al programa de instalación dónde instalar CentOS. Esto se hace mediante la definición de los puntos de montaje para una o más particiones de disco. Es necesario también crear y/o eliminar particiones en este punto.

La herramienta de particionamiento usada en CentOS será el DISK DRUID. Con la excepción de ciertas situaciones “esotéricas”, DISK DRUID normalmente mantiene los requisitos de particionamiento de una instalación normal de CentOS. DISK DRUID ofrece una representación gráfica de nuestro(s) disco(s) duro(s).

A menos que tenga una buena razón para hacer lo contrario, se recomienda que cree las siguientes particiones para los sistemas basados en x86.

Una partición swap

(de al menos 256 MB) Las particiones swap (de espacio de intercambio) son utilizadas para apoyar a la memoria virtual. En otras palabras, los datos son escritos a una partición swap cuando no hay suficiente memoria RAM para almacenar los datos que su sistema esta procesando.

La selección de la cantidad correcta de espacio de intercambio va a depender de varios factores, incluyendo los siguientes (en orden descendente de importancia):

- Las aplicaciones ejecutándose en la máquina.
- La cantidad de RAM física instalada en la máquina.

Ø La versión del sistema operativo.

El espacio de intercambio debe ser igual a 2 veces la RAM física, hasta un máximo de 2 GB de RAM física y luego 1 vez la RAM física para cualquier cantidad sobre los 2 GB, pero nunca menos de 32 MB. La partición debe ser de tipo swap. La creación de una partición swap grande puede ser de gran ayuda si planea actualizar su RAM posteriormente.

Si el esquema de particionamiento requiere una partición swap más grande que 2 GB, debería crear una partición swap adicional. Por ejemplo, si necesita 4 GB de swap, es recomendable crear dos particiones swap de 2 GB. Si tiene 4 GB de RAM, debería de crear tres particiones swap de 2 GB. Red Hat Enterprise Linux permite hasta 32 archivos swap.

Una partición /boot/ (100 MB). La partición montada en /boot/ contiene el kernel del sistema operativo (que permite que su sistema inicie Red Hat Enterprise Linux), así como los archivos utilizados durante el proceso de arranque. Debido a las limitaciones de muchas BIOS de computadores, es una buena idea la creación de una pequeña partición para guardar estos archivos. Para la mayoría de los usuarios es suficiente una partición boot de 100 MB.

Una partición root (500 MB a 5.0 GB). Aquí es donde se localiza / (el directorio raíz). En esta configuración, todos los archivos (excepto aquellos almacenados en /boot) están en la partición raíz. Una partición raíz de 500 MB le permite el equivalente de una instalación mínima, mientras que una partición raíz de 5.0 GB le permitirá realizar una instalación completa, seleccionando todos los grupos de paquetes.

Configuración del gestor de arranque

El siguiente paso en el proceso de instalación es la configuración del gestor de arranque. El gestor de arranque es el primer software que se ejecuta cuando arranca la computadora. Es responsable de la carga y de la transferencia del control al kernel. El kernel, por otro lado, inicializa el resto del sistema operativo.

GRUB (GRand Unified Bootloader), que se instala por default, es un gestor de arranque muy potente ya que puede cargar una gran variedad de sistemas operativos.

Todas las particiones que se pueden arrancar aparecen en una lista, incluso las particiones que usan otros sistemas operativos. La partición que contiene el sistema de ficheros root del sistema tiene la Etiqueta de CentOS-4-i386 (para GRUB) o Linux. Las otras particiones puede que también tengan etiquetas de arranque.

De forma opcional podemos asignar una contraseña al gestor de arranque, esto es útil cuando no hay un buen mecanismo de control de acceso físico a nuestro equipo.

Configuración de la red

El programa de instalación automáticamente detecta los dispositivos de red que se tiene y los muestra en una lista. Una vez que haya seleccionado el dispositivo de red, se puede elegir configurar la dirección IP y la Máscara de red del dispositivo a través de DHCP (o manualmente si no hemos seleccionado DHCP) y se puede también seleccionar activar el dispositivo en el momento de arranque.

Para cambiar la configuración de red después de la instalación, usar el comando SYSTEM-CONFIG-NETWORK.

Configuración del firewall

CentOS ofrece la protección vía firewall (Firewall es un mecanismo de control de acceso entre el equipo y la red, determina que recursos de nuestro equipo están accesibles para los usuarios remotos) para una seguridad mejorada del sistema. Un firewall bien configurado puede aumentar significativamente la seguridad del sistema. Para cambiar la configuración después de la instalación, tenemos disponible la herramienta de administración SYSTEM-CONFIG-SECURITYLEVEL.

Adicionalmente, podemos configurar SELinux (Security Enhanced Linux) durante la instalación. La implementación de SELinux en CentOS está diseñada para mejorar la seguridad de varios demonios de servidores a la vez que minimiza el impacto de las operaciones cotidianas de su sistema, proporciona permisos de forma granulada para todos los sujetos (usuarios, programas y procesos) y objetos (archivos y dispositivos). De forma segura puede otorgar solamente los permisos que una aplicación necesita para funcionar.

Están disponibles tres estados para seleccionar durante el proceso de instalación:

Ø Inhabilitado. Si no desea activar los controles de SELinux en este sistema. La configuración Inhabilitado hace que la máquina no utilice las políticas de seguridad.
Ø Atención. Para que se le notifique de cualquier detalle. El estado Atención asigna etiquetas a los datos y programas y los registra, pero no hace cumplir ninguna política. El

estado Atención es un buen estado de comienzo para los usuarios que eventualmente desean activar completamente SELinux, pero primero quieren ver los efectos que tendrán las políticas en su operación general del sistema.

- Ø Activo. Si se desea que SELinux actúe en un estado completamente activo. El estado Activo hace cumplir todas las políticas, tal como negar el acceso a los usuarios no autorizados para ciertos archivos y programas, para protección adicional del sistema

Selección del soporte del idioma. Se pueden instalar y soportar múltiples idiomas para usarlos en nuestro sistema. Debemos seleccionar un idioma para usarlo como idioma por defecto. El idioma por defecto es aquel que será utilizado por el sistema una vez que la instalación se haya completado. Típicamente, el idioma por defecto es el mismo que seleccionó durante la instalación. Para cambiar la configuración del idioma una vez finalizada la instalación, usar el comando SYSTEM-CONFIG-LANGUAGE.

Configuración del huso horario. Es importante seleccionar el huso horario que corresponda a la ciudad más cercana a la ubicación física de nuestro equipo. Para cambiar la configuración del huso horario una vez finalizada la instalación, usar el comando SYSTEM-CONFIG-DATE.

Configuración de la contraseña de root

La configuración de la cuenta y la contraseña de root es uno de los pasos más importantes durante la instalación. La cuenta root es usada para instalar paquetes, actualizar RPMs y realizar la mayoría de las tareas de mantenimiento del sistema. El usuario root (también conocido como súper usuario) posee acceso completo al sistema; por este motivo, firmarse al sistema como usuario root es aconsejable hacerlo sólo para ejecutar el mantenimiento o administración del sistema.

La contraseña de root debe de tener al menos ocho caracteres y no aparecerá en la pantalla cuando la teclee. Deberá introducirla dos veces; si las dos contraseñas no coinciden, el programa de instalación nos pedirá que la tecleemos de nuevo.

Las contraseñas mejores son aquéllas que mezclan números con letras mayúsculas, minúsculas y caracteres especiales que no formen palabras contenidas en diccionarios. Recordemos que la contraseña es sensible a las mayúsculas y minúsculas. Se recomienda que nunca escriba su contraseña; pero si la escribe en un papel guárdelo en un lugar seguro.

Selección de grupos de paquetes

Primero, aparecerá la pantalla Selección de grupos de paquetes que detalla el conjunto de paquetes que por default se seleccionan de acuerdo al tipo de instalación seleccionada con anterioridad. Podemos seleccionar grupos de paquetes, los cuales agrupan componentes de acuerdo a una función (por ejemplo, Sistema X Windows y Editores), paquetes individuales o una combinación de los dos.

Seleccionemos los componentes que deseamos instalar. Al seleccionar Todo (al final de la lista de componentes) se instalarán todos los paquetes incluidos con CentOS. Una vez seleccionado un grupo de paquetes, podemos hacer clic en Detalles para visualizar los paquetes que se instalarán por defecto y los paquetes opcionales que deseamos eliminar o añadir a ese grupo.

Listo para la instalación

En este paso podemos arrepentirnos sin causar daños a nuestro equipo, una vez dando clic en aceptar comenzara la copia de los archivos CentOS a nuestro disco.

El instalador nos informará de los discos que se necesitarán durante la copia, tenemos que verificar que los poseemos. El proceso de instalación de paquetes puede durar de 15 min a 1 hr, dependiendo la cantidad de paquetes que hayamos elegido y la velocidad de nuestro equipo, durante este proceso podemos monitorear el avance. Durante el proceso de instalación podemos hacer uso de las consolas virtuales.

Una vez terminado el proceso de instalación el equipo se reiniciará, es necesario remover el CD de instalación de la unidad de CD-ROM.

Ambiente Grafico de Unix, Sistema X Windows

Configuración Sistema X Windows

CentOS provee la herramienta SYSTEM-CONFIG-DISPLAY. Los archivos relacionados a X11 residen principalmente en dos ubicaciones:

- Ø /usr/X11R6/: Contiene el servidor X y algunas aplicaciones cliente así como también archivos de cabecera X, bibliotecas, módulos y documentación.
- Ø /etc/X11/: Contiene archivos de configuración para aplicaciones cliente y servidor de X.

Mientras que el corazón de CentOS es el kernel, para muchos usuarios, la cara del sistema operativo es el entorno gráfico proporcionando por el Sistema X Windows, también llamado simplemente X.

En el mundo UNIX, los entornos de ventanas han existido desde hace décadas, siendo éstos precursores de muchos de los utilizados en los sistemas operativos actuales. A través de los años X se ha convertido en el entorno gráfico (GUI) predominante para sistemas operativos del tipo UNIX.

El entorno gráfico para CentOS es suministrado por la Fundación X.Org, una implementación de código abierto creada para manejar el desarrollo y la estrategia para el sistema X y sus tecnologías asociadas. X.Org es un proyecto de gran escala que se apoya en un gran número de desarrolladores en todo el mundo. Presenta una amplia gama de soporte para diferentes dispositivos de hardware y arquitecturas, así como la posibilidad de ejecutarse en diferentes sistemas operativos y plataformas. Este lanzamiento de Red Hat Enterprise Linux incluye específicamente el lanzamiento X11R6.8 del sistema X Windows.

El sistema X Windows utiliza una arquitectura cliente-servidor. El servidor de X (el binario Xorg) escucha por conexiones desde las aplicaciones cliente X a través de la red o una interfaz local de loopback. El servidor gestiona la comunicación con el hardware, que puede ser una tarjeta gráfica, un monitor, un teclado o un ratón. Las aplicaciones cliente de X existen en el espacio del usuario, creando una interfaz gráfica del usuario (GUI) y pasando peticiones al servidor de X.

Una vez que un servidor X se esté ejecutando, las aplicaciones cliente X pueden conectarlo y crear una GUI para el usuario. Un rango de GUIs está disponible con Red Hat Enterprise Linux, desde el rudimentario Administrador de pestañas de ventanas hasta un entorno de escritorio altamente desarrollado, interactivo como GNOME, con el que la mayoría de los usuarios de Red Hat Enterprise Linux están familiarizados. Para crear lo último, una GUI más avanzada, se deben conectar dos clases principales de aplicaciones clientes X al servidor X: un entorno de escritorio y un gestor de ventanas.

Entornos de escritorio

Un entorno de escritorio une diferentes clientes de X, los cuales cuando se usan juntos crean un ambiente de usuario gráfico común y una plataforma de desarrollo. Los entornos de escritorio tienen características avanzadas las cuales permiten a los clientes X y a otros procesos en ejecución, comunicarse unos con otros a la vez que se permite a todas las aplicaciones escritas para funcionar en ese ambiente a que realicen tareas avanzadas, tales como operaciones de arrastrar y soltar. CentOS proporciona dos entornos de escritorio:

- GNOME.- Es el entorno de escritorio por defecto en Red Hat Enterprise Linux basado en el conjunto de herramientas gráficas GTK+ 2.
- KDE.- Un entorno de escritorio alternativo basado en el conjunto de herramientas gráficas Qt 3.

Ambos entornos GNOME y KDE tienen aplicaciones de productividad avanzadas, tales como procesadores de palabras, hojas de cálculo y navegadores Web así como herramientas para personalizar la apariencia de la GUI.

Gestores de ventanas

Los gestores de ventanas son programas clientes de X que son o parte del entorno de escritorio o en algunos casos, independientes. Su propósito principal es controlar la forma en que las ventanas gráficas son posicionadas, redimensionadas o movidas. Los manejadores de ventanas controlan las barras de títulos, el comportamiento del foco, los vínculos del botón del ratón y teclas especificadas por el usuario. Se incluyen cuatro gestores de ventanas con CentOS:

- Kwin.- Es el manejador por defecto para el entorno KDE. Es eficiente y soporta temas personalizados.
- Metacity.- Es el manejador defecto del entorno GNOME. Es simple y eficiente y también soporta temas personalizados.
- Mwm.- Es un gestor básico independiente. Puesto que está diseñado para ser un gestor que se ejecuta de forma independiente, no se debería utilizar en conjunto con los entornos de escritorios GNOME o KDE.
- Wm.- El minimalista Administrador de pestañas de ventanas, el cual proporciona el conjunto de herramientas más básicas de cualquier gestor de ventanas y puede ser usado bien sea de forma independiente o con un entorno de escritorio. Es instalado como parte de la versión X11R6.8.

GNOME

GNOME es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Windows, se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

El proyecto GNOME (GNU Network Object Model Environment) surge en agosto de 1997 como proyecto liderado por Miguel de Icaza para crear un entorno de escritorio completamente libre para sistemas operativos libres, en especial para GNU/Linux. Desde el principio, el objetivo principal de GNOME ha sido proporcionar un conjunto de aplicaciones amigables y un escritorio fácil de utilizar.

En esos momentos existía otro proyecto anterior con los mismos objetivos, pero con diferentes medios: KDE. Los primeros desarrolladores de GNOME criticaban a este proyecto por basarse en la biblioteca de controles gráficos Qt por no ser compatible con los fundamentos del software libre. Años más tarde los problemas de licencia de Qt se han resuelto y estas críticas han cesado. Sin embargo, los dos proyectos siguen rumbos tecnológicos distintos y se hacen una competencia amigable.

Como con la mayoría de los programas GNU, GNOME ha sido diseñado para ejecutarse en toda la gama de sistemas operativos de tipo Unix con X Windows y especialmente pensado para GNU/Linux. Desde sus inicios se ha utilizado la biblioteca de controles gráficos GTK, originalmente desarrollada para el programa The GIMP.

A medida que el proyecto ha ido progresando en los últimos años, los objetivos del mismo se han extendido para tratar una serie de problemas en la infraestructura Unix existente. Actualmente el proyecto evoluciona bajo amparo de la Fundación GNOME.

Cuando entramos a sesión en CentOS aparece un escritorio similar al de la figura

Para salir del sistema basta con dar clic sobre el icono de CentOS y seleccionar la opción terminar sesión. Una ventana de dialogo aparece para confirmar la que deseamos salir de sesión.

Administración del Sistema Operativo

Responsabilidades del Administrador de Sistemas

La dinámica de la operación requiere operar el sistema de manera diferente a la ordinaria, el cambio de un dispositivo ocasionado por la actualización tecnológica, la instalación de nuevos programas, la falla de algún dispositivo que altere el sistema, se requiera de ajustes, etc.

El administrador del sistema es el responsable de que las operaciones diarias del sistema se encuentren funcionando, proporcionar y mantener acceso a los recursos del sistema. Independientemente de la plataforma informática de que se trate, todos los sistemas operativos proporcionan mecanismos para manipular recursos. Entre estos recursos se encuentran los archivos, las aplicaciones, los periféricos, el ancho de banda, los ciclos de procesador, la memoria y el espacio de almacenamiento.

Cualquier persona puede, si quiere, ejecutar aplicaciones en un sistema UNIX, pero no cualquiera lo instala y lo mantiene funcionando. Para esto es necesario de una persona capacitada que normalmente se le conoce como administrador de sistemas.

Responsabilidad del Administrador

Entre las tareas más comunes tenemos las siguientes:

- Mantenimiento de los usuarios del sistema. Los procesos de agregado y eliminación de cuentas de usuario pueden ser en parte automatizados mediante scripts (programas en UNIX u otro lenguaje a nivel de sistema operativo) ya sea en modo gráfico o de caracteres, pero siempre debe tomarse algunas decisiones. Al crear una cuenta de usuario es preciso determinar la incorporación a grupos de trabajo, lugar del directorio propio, máquinas habilitadas, alias de correo electrónico. Al eliminar una cuenta de usuario los archivos creados por ese usuario deben ser eliminados, respaldados en cinta o transferidos a otro usuario, conservando la información de interés y la responsabilidad individual sobre la misma, así como los recursos del sistema.
- Seguridad. El administrador del sistema debe implementar una política de seguridad y verificar periódicamente que la seguridad de sus sistemas no ha sido violada. Según el tamaño de los sistemas y su conectividad, las precauciones de seguridad puede incluir simples limitaciones de acceso hasta elaborados procedimientos de captura y auditoría. La seguridad es cada vez más crucial y demandante de recursos.
- Agregar y quitar hardware. Al adquirir o transferir de un equipo a otro una parte de hardware puede ser preciso actuar sobre el sistema para asegurar el reconocimiento y uso del equipo. La tarea puede consistir en enchufar una impresora, abrir la máquina para instalar un disco, una tarjeta controladora u otro dispositivo, correr un script para reconocer o inicializar el nuevo hardware o editar complejos archivos de configuración.
- Instalación de nuevo software. El software nuevo debe ser instalado, configurado y comprobado su funcionamiento. Luego, los usuarios deben ser informados de su existencia, ubicación y particularidades de uso. El software no perteneciente al sistema operativo debe instalarse en un lugar aparte, para asegurar que las actualizaciones del sistema operativo no lo sobrescriban.
- Realizar respaldos de información. Una tarea esencial frecuentemente olvidada o realizada con descuido. Es responsabilidad del administrador verificar la realización regular de respaldos, la identificación clara de los medios utilizados, la eficacia de la restauración.
- Monitoreo del sistema. Incluye verificar funcionamiento de correo, servidor web y de noticias, mirar los archivos de registro de eventos y errores (log) para detectar o anticipar fallas, asegurar acceso a todas las subredes, controlar recursos del sistema tales como espacio en disco.
- Detección y reparación de fallas. Ante fallas de hardware o software, el administrador del sistema es responsable del diagnóstico, así como de contactar el servicio técnico, realizar la reparación por sí mismo o delegar en su personal, supervisando en todos los casos la tarea.
- Mantenimiento de documentación local. El uso del sistema lo va cambiando respecto a la instalación primaria. Todos los aspectos en los que el sistema vaya siendo alterado o ampliado deben quedar documentados. La documentación abarca particularidades de configuración del sistema operativo, paquetes de software incorporados, tendido de cables, registros de mantenimiento del hardware, estado de respaldos, procedimientos y políticas de administración.
- Ayuda a los usuarios. Aunque pocas veces figura entre las tareas del administrador de sistema, el apoyo a usuarios es muy frecuente e insume mucho tiempo. Es conveniente disponer de procedimientos y ayudas escritos para poner límite al tiempo destinado a estas tareas.

Administración de Cuentas de Usuario y Grupos

UNIX es un sistema operativo multitarea y multiusuario, por lo que se deben establecer ciertos mecanismos para proteger los datos de cada usuario y que puedan ser compartidos en caso necesario.

UNIX posee un mecanismo de permisos asociados a cada archivo. Este mecanismo permite que los archivos y directorios pertenezcan a un usuario en particular. UNIX también permite que los archivos sean compartidos entre usuarios y grupos de usuarios. El comportamiento por defecto en la mayoría de los sistemas es que todos los usuarios pueden leer los archivos de otro usuario, pero no pueden modificarlos o borrarlos.

Existen diferentes categorías de usuarios en función de sus privilegios:

- Súper usuario o root (UID=0). Es el administrador del sistema. Tiene todos los privilegios.
- Usuarios normales. El resto de usuarios que pertenecen a distintos grupos, los cuales pueden tener una serie de propiedades comunes.

Ø Usuarios especiales. Asignados a tareas específicas por el sistema, generalmente de información o manejo de aplicaciones ya instaladas de uso común. Por ejemplo: mail (se encarga de recoger el correo y repartirlo a los diversos usuarios), lp (se encarga de aceptar trabajos de impresión y mandarlos a la impresora), bin, admin, etc.

La información de las cuentas de los usuarios se almacena en los siguientes archivos:

Archivo	Propósito
/etc/passwd	Usuarios Autorizados
/etc/shadow	Contraseñas cifradas
/etc/group	Grupos existentes en el sistema

El archivo /etc/passwd. Contiene la lista de usuarios en el sistema, una línea por usuario. Cada línea contiene siete campos separados por dos puntos (:).
User1:x:102:10:Cuenta usuario 1: /export/home/user1: /bin/ksh

Nombre	Descripción
login	Cada nombre de usuario debe ser único y debe tener de dos a ocho letras y números. El primer carácter debe de ser una letra y al menos un carácter debe ser una letra minúscula. Los nombres no deben tener guiones bajos ni espacios en blanco
Contraseña Cifrada	El password realmente se encuentra en /etc/shadow, la contraseña se cifra (en realidad no es la contraseña cifrada, es el resultado de cifrar una cadena de 8 ceros usando como llave la contraseña) con un algoritmo llamado DES; debe ser fijada por el comando PASSWD o copiar una contraseña ya cifrada de otra cuenta. Si se coloca un asterisco en este campo, la cuenta no puede accederse hasta que el súper usuario asigne una contraseña.
identificación de usuario UID	Un identificador numérico único de usuario para el sistema. Los UID para usuarios regulares están en el rango de 100 a 60000.
identificación de grupo GID	Identificador numérico único de grupo al cual pueden permanecer usuarios. Los UID para usuarios regulares están en el rango de 100 a 60000
Comentario	Tradicionalmente es el nombre completo del usuario. El comando FINGER interpreta este campo como una lista separada por comas, El comando CHFN permite al usuario cambiar su información propia
Directorio Home	Especifica el directorio en el cual se situará el usuario al entrar en el sistema además puede crear y almacenar sus archivos personales. Si se omite algunas versiones pueden ocasionar errores
Shell de entrada	Al ingresar al sistema, el usuario dispone de un intérprete de comandos, generalmente: /bin/sh, /bin/csh, /bin/bash, /bin/ksh, /bin/tcsh, u otros. Este a su vez lanza un script de inicialización.

Cuentas por default

Nombre o Login	ID de Usuario	Descripción
Root	0	Cuenta de súper usuario, configuradas durante el proceso de instalación, no tiene restricciones y sobrescribe todas las protecciones y permisos. Usada por los administradores para realizar tareas administrativas en el sistema.
Bin	1	Cuenta administrativa que es dueña de la mayoría de los comandos
Daemon	2	Cuenta de sistema que controla los procesos en segundo plano
Adm	3	Cuenta administrativa que es dueña de distintos archivos de sistema

Identificación de los usuarios en el sistema

Comando	Descripción
WHO	Muestra los usuarios que están actualmente en el sistema. Presenta identificador de usuario, terminal en que está conectado, fecha y hora de ingreso al sistema.
WHO AM I	Da información sobre el usuario que está trabajando, indicando su máquina y nombre de usuario, terminal, fecha y hora.
WHOAMI	Presenta sólo el nombre del usuario que está operando
ID USUARIO	Proporciona la identificación del usuario invocante, dando el nombre de usuario y su número (UID), nombre de grupo primario y

	su número (GID), nombres de otros grupos a los cuales pertenece y sus números.
FINGER USUARIO	Proporciona nombre del usuario en el sistema, nombre en la vida real y otros datos del usuario invocante, indicando si está en este momento en el sistema y si tiene correo por leer.

El archivo /etc/shadow. En este archivo se encuentran las contraseñas cifradas, una línea por usuario; solo es visible al súper usuario o root. Provee además información relativa a cambio de contraseñas y expiración de la cuenta Debido a la naturaleza crítica del archivo no es recomendable editarlo directamente. En lugar de eso es recomendable mantener los campos del archivo usando los comandos USERADD o PASSWD. Cada línea contiene nueve campos separados por dos puntos (:).

```
root:LXeoktCoMtwZN:6445::::::
```

Nombre	Descripción
LoginID	contiene el login del usuario
Contraseña	un conjunto de caracteres que representa la contraseña cifrada
Lastchg	Indica el numero de días entre el primero de enero de 1970 y la fecha de ultima modificación de la contraseña: registra la fecha en que el usuario cambió
Min	Contiene el mínimo numero de días requeridos entre cambios de contraseñas
Max	contiene el máximo número de días en que la contraseña es valida antes de que el usuario la tenga que cambiar
Warn	número de días en el que se le advertirá al usuario que su contraseña va a caducar
Inactive	número de días inactivos permitidos antes de que la cuenta sea bloqueada
Expire	Fecha en la que la cuenta expira, una vez que expire, el usuario no se puede volver a loguear
El noveno campo	esta reservado para usos futuros

El archivo /etc/Group.- Cada usuario debe pertenecer a un grupo, el cual es el grupo primario y que esta definida por el GID en el archivo /etc/passwd. Cada grupo puede pertenecer hasta a 15 grupos adicionales conocidos como grupos secundarios, los cuales son especificados en el archivo /etc/group. Cada línea contiene cuatro campos separados por dos puntos (:).

```
bin::2:root,bin,daemon
```

Nombre	Descripción
groupname	nombre asignado a el grupo, los nombres de grupo pueden tener un máximo de 8 caracteres
group-password	Contiene un asterisco o un campo vacío, que por lo regular tenía una contraseña de grupo. Actualmente no es aplicable y esta en desuso
GID	número identificador de grupo, deber ser único en el sistema
username-list	una lista de usuarios separados por comas, que representan que esos usuarios forman parte de ese grupo como grupo secundario

Agregar Usuarios. Puedes agregar cuentas de usuario utilizando el comando USERADD. Este programa crea los registros correspondientes en los archivos /etc/passwd y /etc/shadow. El comando USERADD copia automáticamente el contenido del directorio /etc/skel en el directorio home del usuario. Por convención el nombre del directorio home del usuario es el mismo que su login. La sintaxis es la siguiente:

useradd [-u uid][-g gid][-G gid [,gid,..]][-d dir][-m][-s shell][-c comment] loginname

Opciones	Descripción
-u id	Id único para cada usuario
-g grupo	gid o nombre de grupo predefinido
-G grupo	grupos secundarios para el usuario
-d directorio	ruta hacia el directorio del usuario
-m	crea el directorio en el caso que no exista
-s shell	la ruta al shell que el usuario tendrá predefinido
-c comentario	el nombre del usuario
-e expiración	fecha expiración de la cuenta en el formato (mm/dd/yy)
-f inactivo	número de días inactivos permitidos para una cuenta de usuario

Modificar Usuarios.-Es posible utilizar el comando USERMOD para modificar los componentes de una cuenta de usuario. La sintaxis es la siguiente:
usermod [-u uid [-o]] [-g group] [-G group [, group . . .]] [-d dir] [-m]] [-s shell] [-c comment] [-l newlogname] [-f inactive] [-e expire] login

En general, las opciones para el comando USERMOD funcionan como el comando USERADD, con la excepción de las siguientes opciones:

Opciones	Descripción
-l newlogname	Cambia el login para una cuenta existente
-m mueve	Mueve directorio home del usuario al nuevo lugar especificada con opción –d

Borrar Usuarios. Puedes usar el comando USERDEL para borrar las cuentas de los usuarios del sistema. Este comando también remueve los directorios home de los usuarios y todo su contenido si es que se requiere. La sintaxis es la siguiente:
Userdel –r Cuenta de Usuario

Opciones	Descripción
-r	borra el directorio home del usuario del sistema de archivos

Herramientas para la administración de usuarios.
- Ø /usr/sbin/chpasswd. Lee un archivo que consiste de pares de nombres de usuarios y contraseñas y actualiza cada contraseña de usuario de acuerdo a este.
- Ø change. Cambia las políticas de envejecimiento de contraseñas. También se puede utilizar el comando PASSWD para este propósito.
- Ø Chfn. Cambia la información GECOS de un usuario.
- Ø Chsh. Cambia el intérprete de comandos por defecto de un usuario.

Aunque útiles, estos comandos y scripts rara vez implementan todas las políticas de uso locales. Es aconsejable reescribir o adaptar los scripts ADDUSER y RMUSER para realizar vía scripts todo el proceso de creación y remoción de usuarios.

Agregar Grupos. Como root, puedes crear nuevos grupos en el sistema local usando el comando GROUPADD. Este comando agrega entradas para nuevos grupos en el archivo /etc/group. La sintaxis es la siguiente:
groupadd [-g gid [-o]] groupname

Opciones	Descripción
-g gid	Asigna el id de grupo para el nuevo grupo
-o	Permite que el gid se duplique

Modificar Grupos. Puedes usar el comando GROUPMOD para modificar las definiciones del grupo especificado para alterar el archivo /etc/group. La sintaxis es la siguiente:
groupmod [-g gid [-o]] [-n name] groupname

Opciones	Descripción
-g gid	Especifica el nuevo id de grupo
-o	permite que el gid sea duplicado
-n nombre	especifica un nuevo nombre para el grupo

Borrar Grupos. Puedes usar el comando GROUPDEL para borrar un grupo del sistema y lo borra su entrada del archivo /etc/group. La sintaxis es la siguiente:
groupdel groupname

Herramientas para la administración de grupos.
- Ø Gpasswd. Cambia la membresía de un grupo y configura la contraseña para impedir a los usuarios que no sean miembro pero que conozcan la contraseña, unirse al grupo. También se utiliza para especificar administradores de grupos.
- Ø /usr/sbin/grpck. Verifica la integridad de los archivos /etc/group y /etc/gshadow.

Edición de archivos
Si estas herramientas existen en el sistema, la edición de /etc/passwd debe hacerse con VIPW, que bloquea el archivo para evitar interferencias si un usuario está cambiando su contraseña simultáneamente. Se insiste en la necesidad absoluta de mantener coordinados los archivos /etc/shadow y /etc/passwd. La edición de /etc/group se hace con VIGR. Si el

sistema provee scripts para crear, modificar o borrar usuarios, éstos se ocupan de mantener la coherencia y bloquear los archivos antes de modificarlos.

Editar /etc/Group, si bien no es necesario asignar el usuario a su grupo por defecto conviene igualmente hacerlo para tener actualizada la lista de integrantes del grupo. Sí debe editarse /etc/group para incluir al usuario en grupos secundarios

Fijar contraseña inicial. El súper usuario puede cambiar en cualquier momento la contraseña de cualquier usuario mediante el comando PASSWD nombre usuario. Un usuario común puede cambiar su propia contraseña digitando solamente PASSWD. El comando PASSWD suele exigir un mínimo de largo, caracteres de puntuación, números, mezcla de mayúsculas y minúsculas, para alcanzar un cierto nivel de inviolabilidad. No debe dejarse nunca una cuenta sin contraseña, especialmente en ambientes de red o con acceso a Internet.

```
passwd
```

Pide la vieja contraseña y luego la nueva; la nueva contraseña deberá ingresarse dos veces, para evitar posibles errores de digitación.

Caducidad de contraseñas. Los UNIX modernos disponen de una facilidad para obligar al usuario a cambiar su contraseña periódicamente.

Crear directorio propio del usuario

La siguiente secuencia de comandos crea un directorio para el usuario jperez:

```
mkdir /export/home/jperez
chown jperez /export/home/jperez
chgrp usuarios /export/home/jperez
chmod 700 /export/home/jperez
```

En ambientes menos restrictivos, los permisos de los directorios propios pueden ser 755, lo que permite a todos los usuarios del sistema ver los directorios personales. Los usuarios siempre pueden limitar el acceso sobre los subdirectorios contenidos en su directorio personal.

Fijar cuotas. Si en el sistema se utilizan cuotas para controlar el uso de disco, es preciso fijar la cuota para el nuevo usuario con el comando EDQUOTA. Este comando admite diversos parámetros para fijar límites en el uso de disco, pero en la creación de usuarios es más frecuente usarlo en modo prototipo, copiando el perfil de cuota de algún usuario tomado como modelo: EDQUOTA -p usuario-modelo nuevo-usuario

Inhabilitar cuentas. Cuando una cuenta debe ser temporalmente inhabilitada, por ejemplo por ausencia del titular, se solía colocar simplemente un asterisco en el lugar de la contraseña en el /etc/shadow.

Pseudo logins. Son cuentas de usuario que no corresponden a personas. Es el caso de bin y daemon. Es posible crear otras cuentas de pseudo login, como shutdown, halt, who; estas cuentas tienen como shell un programa que sólo ejecuta el comando indicado.

Archivos de inicialización

Es posible configurar los archivos de inicialización y ubicarlos dentro del directorio HOME de cada usuario. El primer trabajo para modificar un archivo de inicialización es definir las características del ambiente de trabajo para el usuario, rutas en donde puede buscar los comandos, variables de ambiente, etc.

Solo el dueño de los archivos o el usuario root pueden modificar el contenido de los archivos.

Archivos de inicialización de shells:

Shell	Archivo de inicialización de todo el sistema	Archivo de inicialización del usuario al iniciar sesión	Archivo de inicialización del usuario al iniciar un nuevo shell después de tener sesión	Ruta del Shell
Bourne	/etc/profile		$HOME/.profile	/bin/sh
BASH	/etc/profile	$HOME/.bash_profile $HOME/.bash_login $HOME/.profile	$HOME/.bashrc	/bin/bash

Estos archivos permiten al usuario configurar el entorno de su cuenta automáticamente cuando entra en el sistema, cuando arranca un subshell o ejecutar comandos cuando sale del sistema.

Los nombres de estos archivos son:.bash_profile, .bashrc. Si ninguno de estos archivos existe en el directorio del usuario, /etc/profile es utilizado por el sistema como archivo de configuración de bash. .bash_profile es el más importante. Es leído y los comandos incluidos en el, ejecutados, cada vez que el usuario entra en el sistema. Cualquier cambio hecho en este archivo no tendrá efecto hasta que salgamos y entremos en el sistema de nuevo. Una alternativa para no tener que salir del sistema es ejecutar el comando SOURCE .bash_source.

Bash permite dos sinónimos para este archivo, .bash_login (derivado del C shell) y .profile (derivado del Bourne y Korn shell). Si .bash_profile no existe, el sistema buscara primero .bash_login y luego .profile. Solamente uno de estos archivos es leído, en el caso que existan simultáneamente.

.bashrc es leído cuando el usuario arranca un subshell, escribiendo por ejemplo bash en la línea de comandos. Esto nos permite ejecutar diferentes comandos para la entrada al sistema o para la ejecución de un subshell.

.bash_logout es el archivo leído por Bash, cuando salimos del sistema. Podemos definir, por ejemplo que se borren los archivos temporales creados en nuestra ultima sesión o registrar el tiempo que hemos estado utilizando el sistema. Si .bash_logout no existe, ningún comando será ejecutado a nuestra salida.

Algunos comandos admiten personalización colocando archivos en el directorio propio del usuario. Estos archivos generalmente empiezan con punto, lo que los hace normalmente ocultos y terminan con rc, por “run command”. Algunos ejemplos comunes:

Comando	Archivo	Usos
csh	.login	Fija tipo de terminal, variables de ambiente, opciones para BIFF y MESG.
	.cshrc.	Fija alias de comandos, rutas de búsqueda, valor de umask, cdpath para búsqueda de nombres de archivo; fija variables promtp, history, savehist
	.logout	Imprime recordatorios, borra la pantalla.
sh	.profile	Fija tipo de terminal, variables de ambiente, opciones para BIFF y MESG; fija alias de comandos, rutas de búsqueda, valor de umask, cdpath para búsqueda de nombres de archivo, etc.
bash	.bash profile	Fija tipo de terminal, variables de ambiente, opciones para BIFF y MESG; fija alias de comandos, rutas de búsqueda, valor de umask, cdpath para búsqueda de nombres de archivo, etc.
	.basrh	Fija alias de comandos, rutas de búsqueda, valor de umask, prompt
vi	exrc	Opciones para el editor VI.
emacs	.emacs pro	Opciones y asignación de teclas para el editor EMACS.
mailx	.mailrc	Aliases personales para correo, opciones para el lector de correo.
tin	.newsrc	Especifica grupos de interés para noticias.
xrdb	.Xdefaults	Fija configuración de X11 (sistema de ventanas): tipos de letra, colores, etc.
startx	.xinitrc	Fija ambiente inicial para X11.
xdm	.xsession	Fija ambiente inicial para X11.

Suele haber un conjunto ya preparado de archivos de inicialización, usualmente en /etc/skel; también pueden crearse en /usr/local/lib/skel, editándolos para reflejar configuraciones útiles al usuario corriente.

Ejemplo Archivo .profile

Acción	Comando
Define la variables de búsqueda path	PATH=$PATH:$HOME/bin:/usr/local/bin:/usr/ccs/bin:
Define la ruta del inbox del usuario	MAIL=/var/mail/$LOGNAME
Define el nombre del servidor para el canal de noticias de Usenet	NNTPSERVER=server1
Define la variable de búsqueda de páginas de manual	MANPATH=/usr/share/man:/usr/local/man

Define la impresora por default	PRINTER=printer1
Crea un alias para el comando history	alias h history
Define los permisos de archivos nuevos	umask 022
Da amplitud global a las variables	export PATH MAIL NNTPSERVER MANPATH PRINTER
Corre algún script	source /net/server2/site-init-files/site.login

El Sistema de Archivos

Definiciones

Entenderemos por archivo a un conjunto de caracteres o bytes. El sistema de archivos no impone ninguna estructura sobre el archivo y su contenido no tiene ningún significado para el; el significado depende exclusivamente de los programas que lo manipulan.

La estructura del sistema de archivos es jerárquica, es decir una gráfica dirigida o una estructura arbórea. El directorio principal, llamado raíz, tiene por nombre simplemente '/' que a su vez es el carácter utilizado para separar los nombres de los subsiguientes directorios.

Al ser GNU/Linux un sistema operativo multiusuario, tiene también un modo de protección al sistema de archivos, previsto para la privacidad entre usuarios. Este mecanismo esta formado por tres tipos de accesos: de usuario, de grupo y de "otros". Esto quiere decir que el usuario fija los accesos para él, para las personas en su grupo y para el resto de los usuarios. En cada caso se tiene un permiso independiente para lectura, escritura y ejecución. Varios usuarios pueden pertenecer al mismo grupo y un usuario puede, a su vez, pertenecer a varios grupos. El fin de esta caracterización de los usuarios es para permitir y restringir accesos a determinadas partes del sistema.

Un sistema de archivos en GNU/Linux puede contener miles de archivos, cientos de directorios y cientos de enlaces simbólicos, dependiendo de la distribución y de lo que se haya instalado. El siguiente comando:

```
$ ls -F /
```

Es útil para recorrer el sistema de archivos, bajando luego a los subdirectorios.

El proyecto Filesystem Hierarchy Standard es el encargado de normar el nombre y contenido de los directorios más comunes en un sistema GNU/Linux.

Estructura de directorios en GNU/Linux

Directorio	Descripción
/bin	binarios del sistema de uso frecuente
/boot	Contiene archivos estáticos requeridos para arrancar el sistema, tales como el kernel de Linux.
/dev	Contiene entradas del sistema de archivos que representan dispositivos del sistema. Estos archivos son esenciales para el correcto funcionamiento del sistema.
	[subdirectorios propios de System V]
./dsk	Dispositivos de disco
./fd	Dispositivos descriptores de archivo
./kd	Dispositivos de teclado y despliegue
./kmem	Memoria
./null	Dispositivo para descarte de salidas
./osm	Mensajes de error del kernel
./pts	Pseudo ttys; igual que /dev/pts*
./Rask	Dispositivos crudos de disco
./term	Terminales; igual que /dev/tty*
./xt	Pseudo ttys; para capas DMD
/etc	Configuración de paquetes, configuración de sistema
./init.d	Scripts de arranque y detención de programas
./rc?.d	Enlaces a scripts, con K o S (Kill o Start) y número de secuencia para controlar el arranque
./skel	Archivos de inicialización para nuevos usuarios
/export	Directorios de usuarios en sistemas grandes
/home	Objetos relacionados con los usuarios
/lib	Bibliotecas de desarrollo y material de apoyo
/lost+found	Archivos perdidos

/mnt	Punto de montaje de dispositivos externos
/proa	Contiene ‘’archivos” especiales que o bien extraen información del kernel o bien la envían a éste.
/root	Directorio propio para el súper usuario (root)
/sbin	Archivos ejecutables de administración
/tmp	Archivos temporales
/usr	Ejecutables, documentación, referencia
./X11R6	Sistema X-Windows
./bin	Más ejecutables
./doc	Documentos de paquetes de software
./include	Encabezados .h de bibliotecas en C
./info	Archivos de info, información de Linux
./lib	Más bibliotecas en C, información propia de programas
./local	Ejecutables instalados por el administrador
./man	Subdirectorios de páginas del manual
./sbin	Más archivos ejecutables de administración
./share	Compartidos
./src	(source) Código fuente del kernel
/var	Archivos de log, auxiliares, archivos que crecen
./backup	Respaldo de algunos archivos del sistema
./catman	Páginas man ya formateadas
./lock	Control de bloqueos
./log	Archivos de registro de mensajes (log) del sistema
./spool	
./run	Información de procesos (PIDs)

Todos los discos que contienen sistemas de archivos en GNU/Linux tienen la siguiente organización. El bloque 0 no es utilizado por GNU/Linux y contiene a menudo código para el arranque de la computadora. El bloque 1 que es el superbloque, contiene información crítica relativa a la organización del sistema de archivos:
Ø número de i-nodos
Ø número de bloques en disco
Ø inicio de la lista de bloques libres en disco.

La destrucción del superbloque provocará que el sistema de archivos quede inservible. Después del superbloque están los nodos-i (abreviatura de nodos-índice, aunque nunca se les llama de esta forma). Se enumera de 1 hasta cierto máximo. Cada i-nodo tiene una longitud de 64 bytes y describe a un archivo. Un i-nodo contiene la información contable (propietario, bit’s de protección, etc) así como la información suficiente para localizar todos los bloques del disco que contengan los datos del archivos.

Organización del disco en los sistemas.

	Carga inicial al arrancar
A	Cabecera SUPER BLOQUE
B	i-nodes
C	Bloques de información

Después de los nodos-i están los bloques de datos. Todos los archivos y directorios se almacenan aquí. Si un archivo o directorio consta de más de un bloque, los bloques no tienen por qué ser adyacentes en el disco. De hecho, es probable que los bloques de un archivo grande estén diseminados por todo el disco. Las mejoras de Berkeley se diseñaron para reducir esta dispersión.

Un directorio en el sistema tradicional de archivos consta de una colección no ordenada de entradas de 16 bytes. Cada entrada contiene un nombre de archivo (de hasta 14 caracteres arbitrarios) y el número del nodo-i del archivo. Para abrir un archivo en el directorio de trabajo, el sistema sólo lee el directorio, compara el nombre por buscar con cada entrada, hasta que encuentra el nombre o concluye que no está presente. Si el archivo está presente, el sistema extrae el número de nodo-i y lo utiliza como un índice en la tabla nodos-i para localizar el nodo-i correspondiente y traerlo a la memoria. El nodo-i se coloca en la tabla de nodos-i, una estructura de datos en el kernel que conserva todos los nodos-i de los archivos y directorios abiertos en cada momento.

La búsqueda de un nombre absoluto de ruta de acceso como /home/paco/fundamentosSO.pdf es un poco más compleja. En primer lugar, el sistema localiza el directorio / (raíz), que siempre utiliza el nodo-i 2 (el nodo-i 1 se reserva para el manejo de los bloques

defectuosos). Entonces busca la cadena ‘’home” en el directorio /, para obtener el número de nodo-i del directorio /home. Se busca entonces este nodo-i y se extraen los bloques del disco de él, de forma que se pueda leer el directorio y buscar la cadena ‘’paco”. Al encontrar esta entrada se puede tomar el número de nodo-i para el directorio /home/paco. Con el número de nodo-i del directorio /home/paco, se puede leer este nodo-i y localizar los bloques del directorio. Por último se busca fundamentosSO.pdf y se encuentra su número de nodo-i. Así, el uso de un nombre relativo de una ruta de acceso no solo es más conveniente para el usuario, si no que también ahorra una cantidad sustancial de trabajo al propio sistema.

El comando utilizado para visualizar el contenido de un directorio es LS con una serie de opciones. Las opciones a programas y comandos en GNU/Linux típicamente van precedidas por el carácter -, salvo cuando se indique lo contrario. Las opciones sirven generalmente para cambiar el comportamiento de un comando. Por ejemplo para ver el contenido de un directorio en forma somera, basta dar LS:

```
$ ls
1erExamenSO.aux     fundamentosSO.pdf      practica2.pdf
1erExamenSO.tex     fundamentosSO.toc      practica3.dvi
acct.eps            instalacion.tex        practica3.log
admon.tex           lang.eps               practica6.aux
```

O en forma más completa LS -al, donde la opción a indica que queremos todos los archivos (Los archivos cuyo nombre empiece con . son archivos ocultos) y la opción l indica que queremos la versión completa o “larga” de la salida:

```
$ ls -la
total 37836
drwxrwxr-x 4 paco paco 4096    dic  1 19:07  .
-rw-rw-r-- 1 paco paco 265     dic  1 11:54  1erExamenSO.aux
-rw-rw-r-- 1 paco paco 49281   nov 30 11:27 acct.png
drwxrwxr-x 2 paco paco 4096    dic  1 15:03  uxo
-rw-rw-r-- 1 paco paco 3670    dic  1 18:39  uxo.aux
-rw-rw-r-- 1 paco paco 2925753 nov 29 20:28 x86bootloader.eps
```

Las primeras diez columnas indican el tipo de acceso que tiene cada archivo para los diferentes usuarios.

```
          1 1 1    1    1    1   1 1     1
1234567890 1 2    3    4    5   6 7     8
-rw-r--r-- 1 paco paco 2495 dic 1 19:01 fundamentosSO.tex
```

Cada columna está numerada, con los siguientes significados:
1) (d) es un directorio, (l) es un link, No es un archivo “especial”
2) El dueño puede leerlo
3) El dueño puede escribirlo
4) El dueño no puede ejecutarlo
5) Los del grupo (users) pueden leerlo
6) Los del grupo no pueden escribirlo
7) Los del grupo no pueden ejecutarlo
8) Los otros pueden leerlo
9) Los otros no pueden escribirlo
10) Los otros no pueden ejecutarlo
11) Número de ligas
12) Usuario
13) Grupo
14) Tamaño en bytes
15) y 16) Fecha de alteración
17) Hora de alteración
18) Nombre del archivo.

En el caso de los archivos ejecutables, aparece una x para denotarlos. Los caracteres para los archivos “especiales” son: - normales, d directorios, l liga simbólica, b dispositivo por bloques, c dispositivo por caracteres. Para cambiar los atributos de un archivo contamos con la instrucción CHMOD, que nos permite cambiar el modo de acceso del archivo. La sintaxis es:

```
chmod quienes operacion objeto
```

Donde quienes puede ser una combinación de ugo con u para fijar el modo para el usuario, g para el grupo y o para los otros, es decir, para los que no son el usuario y que tampoco pertenecen al grupo. La operación es + para añadir y - para retirar el modo a los grupos seleccionados y modo puede ser cualquier combinación de rwx.

El segundo comando más empleado en GNU/Linux es MAN, que es el comando que muestra la ayuda de otros comandos. Las opciones más útiles son -k que lista las páginas de manual de los comandos referentes a la siguiente palabra que se dá, por ejemplo:

```
$ man -k change
chdir (2) -  Change working directory
chmod (1) -  Change the access permissions of files
chown (1) -  Change the user and group ownership of files
passwd (1) - Change password
```

Para salir antes de terminar, teclear 'q'. Lista todos los comandos que contienen la palabra CHANGE. La otra opción más usada, sirve como una referencia rápida para saber que acción ejecuta un comando en particular. Es la opción -f. Por ejemplo:

```
$ man -f chdir
chdir (2) - Change working directory
```

Nos indica que CHDIR es el comando para movernos entre directorios.

Si quisiéramos ver el contenido de un archivo podríamos usar el comando CAT. Listemos el contenido de otro archivo, uno común a todos los GNU/Linux, el directorio /etc y el archivo passwd.

```
     $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

Pues resultó que este archivo tiene más líneas de las que caben en la pantalla. Para poder visualizarlo pantalla por pantalla podemos usar un paginador, MORE, con el comando CAT /etc/passwd | MORE. Si ejecutamos este comando veremos que tenemos manera de detener la salida hasta que demos un espacio y cambia de pantalla, Le estamos pidiendo al sistema operativo que la salida producida por el comando CAT se la pase como entrada al comando MORE. Esta técnica se conoce como piping o "entubado". En la introducción hablábamos de que el diseño de GNU/Linux tenía en mente el reunir una colección de pequeños programas muy generales que nos permitan unirlos o conectarlos para realizar tareas más complejas.

Redireccionamiento. Esta es una base muy fuerte de la construcción de herramientas en GNU/Linux, el poder tomar la salida de un programa y dársela como entrada a otro. Para escribir directamente a un archivo, sin posibilidad de regresarnos a la línea anterior, podemos hacer:

```
$ cat > nombre_del_archivo
^D
```

El carácter ^D, significa: presiónese simultáneamente control y D. Es usado para significar fin de entrada (también se usa para salir de la sesión).

Sabemos que con > pasamos la salida de un comando a un archivo, con >> no solo se pasa la salida sino que la agrega al final del archivo, con < tomamos el contenido de un archivo y se lo pasamos a un comando o programa y con | pasamos la salida de un programa a otro programa. Con ECHO despliega un mensaje a la salida estándar.

Crear y borrar archivos. El comando TOUCH se usa para crear archivos sin contenido. Para eliminar archivos se usa el comando RM y una forma interactiva RM - I

Navegando por el sistema de Archivos. Hemos mencionado que el sistema de archivos es jerárquico y que podemos movernos dentro de él. Una de las herramientas indispensables para navegar en el sistema de archivos es PWD que nos dice en que lugar de la jerarquía nos encontramos. Para crear un directorio se utiliza la instrucción MKDIR.
Para recorrer los directorios o cambiar el directorio actual se utiliza el comando CD. Se puede usar en forma relativa o absoluta. En el primer caso basta con dar el nombre de un subdirectorio a partir de donde estamos. Por ejemplo, podemos usar CD .. con lo que nos pasaríamos al directorio padre de donde estamos o . (punto) que hace referencia al directorio en donde estamos. En el segundo caso, el de directorios absolutos, nos podemos mover dando la trayectoria completa usando (/) para separar los directorios. A la trayectoria que se ha de recorrer para llegar hasta un determinado directorio se le conoce como path.

Existen tres maneras de regresarse al directorio original de trabajo. La primera es usando la variable de ambiente $HOME que tiene definido el valor de éste. La segunda es usando una pseudo variable que también tiene como definición a nuestro directorio de entrada: CD . La tercera es la más sencilla: simplemente usando cd sin ningún argumento, nos regresa a nuestro directorio "home" que es como se le conoce comúnmente.

Habrá ocasiones en que será necesario eliminar un directorio o un archivo. De hecho, para poder eliminar un directorio, debe de estar vacío o en otras palabras, no contener ningún archivo. Para borrar un archivo existe la instrucción:
rm [-i] [-f] filename

Opciones	Descripción
-i	antes de borrar el archivo nos pida confirmación
-f	forzamos el borrado de un archivo aún en situaciones especiales
-r	borra subdirectorios

Para eliminar un directorio —que debemos de recordar que tiene que estar vacío— existe el comando RMDIR.

Al decirle que borre DirName/* le decimos que borre todos los archivos contenidos dentro de DirName. El carácter * es un comodín que significa cualquier carácter que se repite 0 o más veces.

Cuando se usan más de una opción en un comando se pueden se pueden aglutinar todas con un sólo -.

Cuando se borra un archivo en GNU/Linux, se libera el espacio en disco que éste ocupaba y queda disponible para que el sistema operativo lo utilice en la creación de otro archivo. Como GNU/Linux es un sistema operativo multitareas y multiusuarios, mientras nosotros borramos un archivo, puede ocurrir que el sistema operativo u otro usuario estén creando uno nuevo, de tal manera que de inmediato se ocupe el espacio donde estaba nuestro archivo. Esto quiere decir que es muy probable que en cuanto nosotros borremos el archivo, el área de disco que ocupaba sea utilizada para escribir otro. O sea, que una vez borrado quizá nunca más lo volveremos a ver. En la mayoría de los casos, es prácticamente imposible recuperar un archivo una vez que ha sido borrado.

Renombrado de archivos. Ahora bien, si lo que necesitamos es cambiar el nombre de un archivo, existe la instrucción MV cuya función real es mover un archivo de un lugar a otro. Si a MV se le da una lista de archivos y al final el nombre de un directorio, lo que hará es mover esos archivos al directorio especificado.

Ligas a archivos. Cuando dos grupos de personas desean trabajar sobre los mismos archivos se pueden crear ligas. El comando es:
ln /home/pedro/trabajo/proyecto.puente proyecto.puente

A continuación se le debe dar los permisos apropiados.
```
chgrp puente proyecto.puente
chmod u=rwx proyecto.puente
chmod g=rwx proyecto.puente
chmod o-rwx proyecto.puente
```

Esta liga se muestra como contador en la descripción de la lista de archivos.
```
     $ ls -la
total 37836
drwxrwxr-x 4 paco paco 4096 dic 1 19:07  .
```

Se pueden borrar las ligas a los archivos pero cuando se borra la última se elimina el archivo.

Cómo revisar archivos. Así como tenemos la instrucción CAT para revisar el contenido de un archivo y la instrucción MORE para realizarlo de una manera pausada, tenemos las instrucciones HEAD Y TAIL que permiten examinar nada más el principio y el final de un archivo respectivamente. Ambos pueden llevar como argumento el número de líneas que se desean examinar. Por ejemplo
```
$ head -3 datos
22:58401|10510612211310910909310100100100100207207109014524 2|121
22:46400|10910710710513012612412600000000100005005109014528 2|121
22:46480|14214316214416815812714800200300300203904509014529 1|121

$ tail -3 datos
000:11785|05504304500010411312000000400600400009509120102223 1|222
000:03368|00300300200007107107100000100200200002902520102223 3|122
000:24120|00500500500014513914700000123545424000000000002802 5|321
```

Ahora supongamos que queremos ver de la línea 496 a la línea 500 del mismo archivo:
```
$ head -500 datos | tail -4
091111024|121
20:40000|097028000000120041000000001000000000051014091111032|122
```

23:37429|1151091151151751811731670000020020000490440911111041|121
23:51446|1161101021121741781831700010020010010370390911111062|121
23:34504|1281141191312345642441781861781710010010020010640001|321

Si necesitamos contar los caracteres, palabras o líneas usamos el comando:
$ wc datos
27588 55941 5231568

Opciones	Descripción
-w	Cuenta palabras
-l	Cuenta líneas

Impresión de archivos. Para imprimir un archivo existe la instrucción LPR que además de enviarlo a la impresora, lo pagina o separa por páginas y le añade un encabezado con el nombre del archivo y número de página.

Opciones	Descripción
-#l	Especifica el largo de la página en líneas
-#d	Que se imprima a doble espacio
-o#	Especifica el margen izquierdo

Montaje de Archivos
Un sistema de archivos contiene un directorio principal, subdirectorios y archivos soportados en un área de disco. Pueden adjuntarse otros sistemas de archivos al árbol de directorios principal aplicando sobre un directorio del árbol principal el directorio superior del sistema de archivos que se adjunta. El punto del árbol principal donde se ‘’cuelga” el sistema de archivos adjuntado se llama punto de montaje. Este artificio permite disponer de un árbol único de directorios formado por partes en distintos soportes locales o remotos. El comando MOUNT habilita la operación de “colgar” un sistema de archivos a un directorio del árbol principal.
mount /dev/sda0 /usuarios

Coloca el sistema de archivos almacenado en el disco /dev/sda0 bajo el directorio usuarios; para ver el contenido del sistema de archivos en disco, en lo sucesivo se hará LS /usuarios.

Un sistema de archivos se desvincula del árbol principal con el comando UMOUNT:
umount /usuarios

Desmonta el sistema de archivos anterior del directorio /usuarios, impidiendo el acceso a los archivos contenidos en el disco. Para poder ejecutar UMOUNT no debe haber archivos abiertos, ni usuarios en directorios de la rama a desmontar, ni procesos corriendo cuyos ejecutables residan en él.

La lista de archivos montados en un sistema UNIX figura en el archivo /etc/fstab, /etc/vfstab o /etc/checklist según la variedad de UNIX. Esto permite realizar tareas sobre todos los sistemas de archivos, generalmente en el momento del arranque. Para verificar la integridad de todos los sistemas de archivos según figuran en /etc/fstab (o su equivalente);
umount fsck -p

Monta todos los sistemas de archivos en /etc/fstab (o su equivalente):
mount -a

Si el intento de desmontar un sistema de archivos fracasa por encontrarse éste ocupado, el comando FUSER indicará los números de proceso y los nombres de usuario que lo ocupan. En FreeBSD, FSTAT hace algo similar.

Encendido y Apagado del sistema

El sistema operativo Linux es un sistema complejo; los procesos de arranque y detención de una máquina Linux implican muchas tareas; deben realizarse correctamente si se desea mantener la salud del sistema. No basta con prender o apagar un interruptor. Los procesos de arranque y detención son dependientes de hardware, puede haber diferencias para un equipo en particular.

Encendido

Bootstrapping. El arranque del sistema suele llamarse “booting” o “booteo” en la jerga informática. En el paso inicial durante el arranque no están disponibles los servicios del sistema; éste debe levantarse a sí mismo iniciando todos sus servicios (bootstrapping).

Cuando una máquina se enciende, ejecuta un programa de carga cuyas instrucciones se encuentran almacenadas en ROM. Este programa determina como cargar en memoria el núcleo del sistema operativo (kernel) y comenzar a ejecutarlo. El kernel examina el hardware probando todos los dispositivos conectados, e inicia un proceso llamado init, siempre con identificador de proceso PID 1. Se montan, verifican los sistemas de archivos, se arrancan los demonios del sistema, siguiendo los dictados de una serie de scripts en lenguaje de shell llamados scripts rc (rc = run command). El contenido y estructura de los scripts rc determinan la situación final del sistema.

Arranque automático y arranque manual

La mayoría de los sistemas operativos tienen un modo de arranque manual y otro automático:

- Ø En modo automático. El sistema operativo realiza las tareas correspondientes al proceso de arranque en forma autónoma, sin necesidad de intervención del administrador, ejecutando todos los scripts de arranque e iniciando todos los procesos necesarios para brindar los servicios habituales a los usuarios.
- Ø En modo manual. El sistema operativo ejecuta una primera parte del proceso de arranque pero casi enseguida transfiere el control al administrador. Se ejecutaron sólo unos pocos scripts, hay pocos procesos corriendo, sólo el súper usuario puede acceder al sistema, se está ejecutando en modo monousuario (single-user mode).

En la operativa diaria el sistema arranca en modo automático con sólo encender el equipo. Algunas fallas pueden obligar al arranque en monousuario, una falla en una tarjeta de red o un sistema de archivos corrupto. Es preciso conocer bien el proceso de arranque para configurar el arranque automático de los servicios requeridos o intervenir en caso de falla.

Pasos del proceso de arranque

El proceso de arranque tiene varios pasos que en general son los siguientes:

1) Carga e inicialización del núcleo (kernel).
2) Detección y configuración de dispositivos.
3) Creación automática de procesos base.
4) Intervención del administrador - solo en modo monousuario).
5) Ejecución de scripts de inicialización.
6) Operación en multi-usuario.

El administrador tiene poco control de esta secuencia, pero sí puede alterar los scripts de inicialización, donde se arrancan los procesos capaces de brindar servicios a los usuarios.

Inicialización del núcleo (kernel). El núcleo del Linux es un programa y como todo programa debe cargarse previamente en memoria para poder ejecutarse. El núcleo reside en un archivo llamado unix, vmunix, vmlinuz o similar. Al encender el equipo comienzan a ejecutarse instrucciones en ROM cuyo objeto es transferir a memoria un pequeño programa de arranque (boot program) encargado de cargar el kernel en memoria y comenzar a ejecutarlo. Esta primera parte del proceso, hasta alcanzar la ejecución del kernel, la realiza el hardware de la máquina. Una vez iniciado, el kernel verifica la cantidad de memoria, separa una parte para sí mismo e informa la cantidad de memoria total, lo reservado para sí y lo disponible para procesos de usuario.

Configuración del hardware.

Al comenzar su ejecución el núcleo intenta localizar e inicializar los dispositivos que le hayan sido asignados en su construcción. Prueba estos dispositivos uno por uno, intentando determinar parámetros de funcionamiento no especificados interrogando al propio dispositivo. Los dispositivos no hallados o que no responden son inhabilitados.

Una vez completado este proceso, si se agrega un nuevo dispositivo deberá esperar el próximo arranque del sistema para ser reconocido. Los sistemas Linux suelen venir con uno o más núcleos genéricos donde están preconfigurados los dispositivos más comunes, pero es posible reconstruir el núcleo optimizándolo estrictamente al hardware disponible.

Procesos del sistema.

Una vez completada la inicialización básica el núcleo crea algunos procesos espontáneos, llamados así por no haber sido creados por FORK, el mecanismo habitual en Linux.

Procesos espontáneos en BSD

Swapper	Proceso 0
init	Proceso 1
pagedaemon	Proceso 2

Procesos espontáneas en System V

sched	Proceso 0
init	Proceso 1
	varios manejadores de memoria y procesos del núcleo

En Linux no hay un proceso visible con PID 0, sino varios procesos de manejo además de init, diferentes según la versión del kernel como se observa en la siguiente tabla

Procesos espontáneos en Linux

Init	Proceso 1
init	Varios manejadores de memoria y procesos del núcleo (kflushd, kupdate, kpiod, kswapd).

Intervención del operador (solo en arranque manual)
Si el sistema fue arrancado en modo monousuario el proceso INIT es invocado por el núcleo con un parámetro indicativo, pidiendo la contraseña de súper usuario y arrancando un intérprete de comandos (shell); al finalizar la ejecución del intérprete INIT continúa con el proceso normal de arranque. Digitando Ctrl-D en lugar de la contraseña del súper usuario continúa el arranque sin invocar el shell.

En el shell monousuario el súper usuario puede trabajar como en cualquier sesión, pero solo dispondrá de comandos existentes en la partición raíz, la única montada. Esta partición puede, además, haber sido montada en solo lectura para ser verificada; si /tmp está en la partición raíz, programas necesitados de archivos temporales como VI no funcionarán. Volver a montar la partición raíz en modo lectura escritura puede diferir entre sistemas, pero en general MOUNT / leerá de nuevo /etc/fstab para montar / en el modo habitual. Otros sistemas de archivos, como /usr, pueden ser montados a mano si es necesario. Una falla habitual de arranque son sistemas de archivos con inconsistencias; en este caso, deberá correrse FSCK en forma manual antes de montar el recurso. En el modo automático los sistemas de archivos son verificados como parte del proceso de arranque; no así en monousuario.

Scripts de inicialización (scripts rc)
En BSD, los scripts rc se ubican bajo /etc, con nombres comenzados por ”rc”; pueden invocarse unos a otros. En System V, como es el caso de Linux, los scripts se guardan bajo /etc/init.d, con enlaces hacia ellos en directorios /etc/rc0.d, /etc/rc1.d, ..., correspondientes cada uno a un nivel de arranque.

Operación en multiusuario.
Una vez ejecutados los scripts de arranque el sistema se encuentra operativo, pero para aceptar el ingreso de usuarios (login) INIT debe arrancar un proceso GETTY de escucha en cada una de las líneas de conexión de terminales. Esto completa el proceso de arranque. En los sistemas configurados para usar login en ambiente gráfico INIT arranca estos servicios (XDM, GDM, DTLOGIN u otro).

El proceso INIT sigue desempeñando un rol importante durante todo el período de funcionamiento del sistema, arrancando procesos y recibiendo en herencia procesos sin padre. En BSD INIT tiene solo dos estados, monousuario y multiusuario; en System V existen diferentes estados mono y multiusuario (run levels), cada uno con diferente espectro de recursos y servicios.

Usualmente los scripts rc realizan, entre otras, las siguientes tareas:
Ø Fijar el nombre de la máquina.
Ø Fijar la zona horaria.
Ø Verificar los discos con FSCK (en arranque automático).
Ø Montar los discos del sistema.
Ø Eliminar archivos del directorio /tmp.
Ø Configurar las interfaces de red.
Ø Arrancar los demonios del sistema y los servicios de red.

Scripts rc estilo System V.
El estilo System V es el más usado actualmente. Se definen en él 7 niveles de arranque (run levels), cada uno con un conjunto de servicios particular:

Run Levels en System V

Run Level	Usar este nivel para
Nivel 0:	Sistema bajo, no hay nada corriendo.
Nivel 1 o S:	Monousuario.
Niveles 2 a 5:	Diversos (o idénticos) niveles multiusuario.
Nivel 6:	retranque (reboot), el sistema se detiene y vuelve a arrancar.

Run Levels en Solaris:

Run Level	Init State	Type	Usar este nivel para:
0	Power-down state	Power-down	Dar de baja el sistema operativo, para apagar el equipo sin ningún problema.
s or S	Single-user state	Single-user	Para ir a single user con todos los sistemas de archivos montados y accesibles.
1	Estado administrativo	Single-user	Acceder a todos los sistemas de archivos sin que los usuarios puedan entrar al sistema.
2	Estado Multiusuario	Multiuser	Para operaciones normales. Los usuarios pueden acceder al sistema y a todos los sistemas de archivos. Todos los daemons esta esperando con excepción de los servicios de NFS.
3	Multiusuario con servicio NFS	Multiuser	Para operaciones normales con NFS.
4	Estado Multiusuario especial		Este nivel no esta disponible.
5	Estado de apagado	Power-down	Para apagar el equipo. Si es posible apagar el equipo de manera automática.
6	Estado Reboot	Reboot	Para llevar el equipo al nivel 0 y después rebootear a modo multiusuario. (o cualquier otro nivel definido en el archivo inittab).

Run Levels en CentOS:

Run Level	Descripción:
0	Halt
1	Modo single user
2	Multusuario, sin NFS (Igual que el nivel 3 pero sin red)
3	Modo Multiusuario con red
4	No definido
5	X11
6	Reboot

Como nivel multiusuario se usa únicamente el nivel 2 o el 2 y el 3 para configuración sin y con red, por ejemplo; los niveles 4 y 5 no suelen ser usados. Los niveles 1 y S difieren según los sistemas. Los niveles 0 y 6 no son en realidad estados de funcionamiento, sino transitorios. El nivel 1 es el monousuario por excelencia; el S fue creado para pedir la contraseña del súper usuario.

Los niveles se definen en el archivo /etc/inittab. Aunque el formato varía, el propósito de este archivo es definir los comandos a correr en cada nivel; uno de ellos define el nivel por defecto que ha de alcanzar el sistema. Los scripts se ejecutan pasando ordenadamente por cada nivel, creciendo en el arranque y decreciendo en la detención. Los scripts de cada nivel detienen los servicios correspondientes a niveles superiores y arrancan los servicios correspondientes al nivel propio. No suele ser necesario tocar el archivo /etc/inittab; el control puede realizarse totalmente a través de los scripts rc correspondientes a cada nivel, confiando en el sistema para la invocación ordenada de los scripts rc de cada nivel según el directorio en que se encuentran.

Los scripts colocados en /etc/init.d manejan cada uno un demonio, servicio o aspecto particular del sistema. Todos los scripts deben entender al menos los parámetros start (arrancar), stop (detener); muchos entienden también restart, un stop seguido de un start. Esto permite al administrador gobernar un servicio manualmente con comandos como:

```
/etc/init.d/sshd start
/etc/init.d/sshd stop
```

El conjunto de servicios a detener y arrancar en cada nivel se controla disponiendo de un directorio para cada nivel y estableciendo en él enlaces hacia los scripts en init.d, según la siguiente convención de nombres:

```
[K|S][0-9][0-9]nombre_script
```

Nombre de enlace iniciado en K invoca el script en init.d con parámetro ”stop”. Nombre de enlace iniciado en S invoca el script en init.d con parámetro ”start”, el número de 2 cifras que sigue a K o S indica el orden de ejecución, sigue el nombre del script en init.d. Ejemplo: S34named, K29bind.

Los directorios para cada nivel siguen la convención de nombres rcN.d, donde N es el número de nivel. Ejemplos: rc0.d, rc2.d, rcS.d. Cuando el sistema está subiendo, se ejecutan los scripts S; cuando está bajando se ejecutan los scripts K, siempre en el orden definido por los números en los nombres de sus enlaces simbólicos. Los comandos:
ln -s /etc/init.d/sshd /etc/rc2.d/S99sshd
ln -s /etc/init.d/sshd /etc/rc2.d/K25sshd

Crean los enlaces simbólicos necesarios para arrancar y detener el servicio sshd en el nivel multiusuario.

Para determinar el estado en el que se encuentra Linux: WHO -r

Problemas de arranque. Las dificultades de arranque pueden deberse a diferentes causas. Antes de diagnosticar un problema de hardware es preciso descartar fehacientemente errores en la configuración del software; muchos de esos errores sugieren fallas de hardware a los ojos inexpertos. Un mensaje de error específico proveniente directamente de un dispositivo es un buen indicador de falla de hardware (error de memoria, error de controlador de disco). Siempre es conveniente verificar aspectos de instalación tales como:

- Ø Alimentación de corriente para todas las partes del equipo: gabinete principal, gabinetes externos con discos, cinta u otros periféricos.
- Ø Alimentación de dispositivos internos (discos, disqueteras, unidades de cinta, CDROM).
- Ø Conexiones entre diferentes dispositivos (cables de señal de discos, disqueteras, puertos).
- Ø Conexiones de red, sobre todo si la máquina depende de recursos externos (estación sin disco, sistemas de archivos montados de otro servidor de red).
- Ø Si existen, revisar e interpretar las luces indicadoras de estado de los dispositivos.
- Ø Si la máquina comienza el arranque, verificar la aparición de todos los dispositivos conectados; cada uno emite un mensaje de una línea al ser reconocido. Un mensaje de error o la falta de detección de un dispositivo conectado correctamente puede indicar una falla de hardware en ese dispositivo, en su tarjeta controladora o en la plaqueta principal.
- Ø Cuando sea posible, usar programas de diagnóstico; los hay para dispositivos independientes del sistema operativo. Las máquinas de hardware propietario disponen de rutinas de diagnóstico de hardware en ROM. En las computadoras personales el BIOS ejecuta en el arranque una prueba de auto verificación llamada POST (Power On Self Test) que prueba la memoria, presencia de discos y otros elementos básicos; los periféricos suelen traer programas para diagnóstico (tarjetas de red, de video, impresoras).
- Ø Al detectarse una falla conviene dejar el equipo apagado unos 30 segundos y luego arrancarlo, para asegurarse de llevar el hardware a un estado conocido.

Problemas con el gestor de arranque del sistema operativo
En las máquinas de arquitectura propietaria existe un programa de carga del sistema operativo almacenado en ROM. Una falla común es la alteración o borrado de una variable de ambiente almacenada en memoria no volátil. En este y otros casos es preciso seguir las indicaciones del fabricante para reponer el arranque.

Los sistemas Linux para computadores personales vienen programas de carga tales como GRUB para Linux o BOOTEASY para FreeBSD. Estos gestores permiten arrancar electivamente Linux u otros sistemas operativos instalados en diferentes particiones de un mismo disco. Una falla, alteración o borrado del programa gestor obliga a arrancar el sistema con un medio alternativo (disquete, CD) y reinstalar el programa de carga o corregir y reponer su configuración correcta.

Las fallas de carga pueden darse en un sistema operativo instalado correctamente; la reposición del esquema de arranque recupera el sistema completamente.

Problemas en el sistema de archivos
Un sistema de archivos puede fallar por razones físicas (superficie dañada, disco roto) o lógicas (inconsistencias en las estructuras de almacenamiento). En caso de falla física habrá que cambiar el disco o al menos realizar una detección de bloques defectuosos para evitar su uso, crear nuevamente el sistema de archivos y reponer desde respaldos. Una falla lógica puede ser reparable con FSCK, eventualmente con alguna pérdida; de no ser así, será preciso recrear el sistema de archivos y reponer desde respaldo. La verificación deberá realizarse arrancando en modo monousuario, con el sistema de archivos en falta desmontado y sobre el dispositivo, ejecutando FSCK reiteradamente en modo forzoso hasta no obtener errores.

Si la falla es en el sistema de archivos raíz el sistema no podrá arrancar. Si se dispone de una partición de arranque alternativa, puede levantarse el sistema desde ella y reponer el servicio rápidamente. Según los sistemas, esto puede hacerse con comandos de ROM o del

programa de carga; es preciso tener muy clara esta secuencia y haber probado el arranque al crear la partición alternativa, así como mantener permanentemente sincronizadas (idénticas) ambas particiones de arranque, mediante copia cruda con DD. Estas particiones deben estar en discos distintos para obtener una confiabilidad razonable. En ausencia de partición de arranque alternativa será preciso reinstalar el sistema operativo, eventualmente reponiendo desde respaldo los archivos de datos.

Una falla lógica en el sistema de archivos raíz puede intentar repararse arrancando el sistema desde un medio alternativo (CD, disquete de rescate o disquete de instalación) y correr FSCK sobre el sistema de archivos raíz, no montado por el arranque desde medio alternativo.

Problemas de configuración del núcleo (kernel). Cuando se genera un nuevo núcleo debe mantenerse siempre un respaldo del núcleo anterior y saber como levantar el sistema con él; un núcleo recientemente generado puede muy bien no funcionar. En Linux es posible generar el nuevo núcleo en disquete, para fines de prueba y recién después copiar la nueva versión sobre la anterior en disco.

Errores en los scripts de arranque. Los errores en los scripts de arranque son la causa más frecuente de falla en el arranque; son también las de más fácil solución: basta arrancar el sistema en monousuario, editar los scripts rc, corregir los errores y continuar con el arranque normal. Es preciso verificar qué editor hay disponible en modalidad monousuario y saber manejarlo; algunos sistemas requieren montar /usr para disponer de VI. El editor ED, poco amigable, suele estar disponible en todos los Linux. En Linux, el disquete de rescate ofrecido por las distintas distribuciones suele tener un conjunto de herramientas razonables para lidiar con estas dificultades; puede dar mejores resultados recurrir al disquete de rescate que intentar el arranque monousuario.

Apagar

Baja del sistema

1). Los sistemas Linux suelen realizar tareas críticas, atender múltiples usuarios o soportar redes de datos; no puede pensarse en reiniciar de nuevo el sistema así como así; es imprescindible pensar primero e intentar solucionar los problemas sin reiniciar la máquina. Es normal para una máquina Linux pasar meses sin detenerse. Solo en casos de agregar un dispositivo, sufrir una falla de hardware, un estado de confusión del sistema o la imposibilidad de acceder, puede pensarse en rearrancar. En suma, cuando realmente no hay otra solución.

Si se han hecho cambios en los scripts de arranque conviene reiniciar el sistema para verificar el funcionamiento correcto de los cambios efectuados sin esperar el próximo inicio.

2) Hay diferentes formas de bajar un sistema o arrancarlo de nuevo:
- Ø Botonazo.
- Ø Usar el comando SHUTDOWN.
- Ø Usar el comando HALT y REBOOT (BSD, Linux, Solaris).
- Ø Usar INIT para cambiar el nivel de ejecución. El comando init le avisa al proceso INIT (El proceso INIT no es lo mismo que el comando INIT). Ejemplo

```
init [S] [1] [-q]
```

Opciones	Descripción
S	Pasa el sistema a modo monousuario en Solaris y HP-UX; en Linux
1	Realiza la misma acción (S solo abriría un nuevo shell). No hay período de gracia ni advertencias
-q	Obliga a init a releer el archivo inittab

3) Ni siquiera en sistemas Linux personales es aceptable bajar el equipo presionando el botón de poder: pueden perderse datos o corromperse lógicamente los sistemas de archivos. Apagar con el botón de poder puede ser inevitable en caso de catástrofe natural o de bloqueo total del sistema. Un equipo Linux tiene que pagarse cuando se ejecuten una de las siguientes tareas:
- Ø Añadir o quitar algún dispositivo.
- Ø Prepararse para una interrupción del suministro eléctrico.
- Ø Hacer actividades de mantenimiento, como respaldos o instalar o remover software.

shutdown

El comando shutdown es el método seguro y completo de apagar un sistema Linux. Dispone de una variedad de opciones, variables según los Linux, pero en general permite fijar anticipadamente el momento del apagado, enviar avisos a los usuarios, decidir si será un apagado, un reinicio o una cambio de runlevel a monousuario. Procede enviando señales de

terminación a todos los procesos en forma ordenada, sincroniza y desmonta los sistemas de archivos, avisa cuando terminó (”System halted”, ”Power down” o similar). Si no aparece este mensaje puede significar que no todas las particiones de discos duros han sido desmontadas y puede llevar a un sistema de archivos corrupto. La sintaxis es la siguiente:

 # shutdown -iinit-state -ggrace-period –y –f [-h now] [-r now]

Opciones	Descripción
-iinit-state	Lleva el sistema al nivel diferente al default. Las opciones son 0, 1, 2, 5 y 6.
-grace-period	Indica el tiempo (en minutos) antes de que el sistema se apage. El tiempo por default son 60 segundos.
-y	Continúa el proceso de apagado sin intervención; de otra manera es necesaria la intervención del administrador para continuar con el proceso de apagado
-n	no sincroniza discos; se usa cuando se ha hecho un FSCK sobre la partición raíz para impedir al kernel sobrescribir el superbloque reparado con el defectuoso que retiene en memoria
-h	Después de apagar todo se detendrá la máquina. Realiza las tareas esenciales mínimas para bajar el sistema ordenadamente: registra la bajada, termina los procesos no esenciales, sincroniza los discos y termina la ejecución
-r	Después de apagar todo se reinicia la máquina
-f	no realiza verificación de discos en el siguiente inicio
Now	Apaga de inmediato

Administración Básica de red

Usando Linux, todas las comunicaciones de red acontecen entre interfaces, que son dispositivos de red conectados al sistema, configurados de un modo determinado y usando un protocolo para intercambiar datos con otros sistemas. Los diferentes tipos de interfaz que existen son tan variados como los dispositivos que los soportan.

Los archivos de configuración para las diferentes interfaces de red y scripts para activarlos o desactivarlos están ubicados en el directorio /etc/sysconfig/network-scripts. Mientras que la existencia de archivos de interfaces particulares puede diferir de sistema a sistema dependiendo del uso, los tres tipos de archivos diferentes que existen en este directorio, archivos de configuración de interfaz, scripts de control de interfaz y archivos de función de red, funcionan conjuntamente para habilitar Linux para el uso de diversos dispositivos de red disponibles.

Antes de revisar los archivos de configuración de interfaz estudiemos los archivos de configuración principales que usa Linux para configurar la red. La comprensión del papel que desempeñan en la configuración de la red es fundamental para configurar el sistema. Los principales archivos de configuración de la red son los siguientes:

- /etc/hosts. Su propósito es resolver los nombres de hosts que no se pueden resolver en otra manera. Se puede usar solamente para resolver nombres de hosts en pequeñas redes sin servidor DNS. Sin tener en cuenta el tipo de red que la computadora use, este archivo contiene una línea que especifica la dirección IP del dispositivo loopback (127.0.0.1) como por ejemplo localhost.localdomain.
- /etc/resolv.conf. Especifica las direcciones IP de los servidores DNS y el dominio de búsqueda. A menos que se haya configurado para algo diferente, los scripts de inicialización de la red llenan este archivo.
- /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.

Servicio xinetd

El demonio xinetd es un superdaemon que controla el acceso a un subconjunto de servicios de red tales como FTP, IMAP y Telnet. También proporciona opciones de configuración específicas al servicio para el control de acceso, registro mejorado, redireccionamiento y control de utilización de recursos.

Cuando un host cliente intenta conectarse a un servicio de red controlado por xinetd, el superdaemon recibe la petición y verifica por cualquier regla de control de acceso wrappers TCP. Si se permite el acceso, xinetd verifica que la conexión sea permitida bajo sus propias reglas para ese servicio y que el servicio no esté consumiendo más de la cantidad de recursos o si está rompiendo alguna regla. Luego comienza una instancia del servicio

solicitado y pasa el control de la conexión al mismo. Una vez establecida la conexión, xinetd no interfiere más con la comunicación entre el host cliente y el servidor.

Archivos de configuración xinetd
Los archivos de configuración para xinetd son los siguientes:

El archivo /etc/xinetd.conf.- Contiene parámetros de configuración generales los cuales afectan cada servicio bajo el control de xinetd. Se lee una vez cuando el servicio xinetd es iniciado, por esto para que los cambios de la configuración tomen efecto, el administrador debe reiniciar el servicio xinetd. Ejemplo del archivo /etc/xinetd.conf:

```
Defaults {
   instances = 60
   log_type = SYSLOG authpriv
   log_on_success = HOST PID
   log_on_failure = HOST
   cps = 25 30
}
includedir /etc/xinetd.d
```

Donde:
- Ø instantes. El máximo número de peticiones que xinetd puede manejar simultáneamente.
- Ø log type. Para usar la facilidad de registro authpriv, el cual escribe las entradas de registro al archivo /var/log/secure.
- Ø log on success. Registra si la conexión es exitosa. Por defecto, la dirección IP del host remoto y el ID del proceso del servidor procesando la petición son grabados.
- Ø log on failure. Registra si hay una falla de conexión o si la conexión no es permitida.
- Ø Cps. Mázimo de 25 conexiones por segundo a cualquier servicio dado. Si se alcanza este límite, el servicio es retirado por 30 segundos.
- Ø includedir. Incluye las opciones declaradas en los archivos de configuración específicos del servicio localizados en el directorio /etc/xinetd.d/.

El archivo /etc/xinetd.d/. Los archivos en el directorio /etc/xinetd.d/ contienen los archivos de configuración para cada servicio manejado por xinetd y los nombres de los archivos que se correlacionan con el servicio. Como sucede con xinetd.conf, este archivo es de sólo lectura cuando el servicio xinetd es arrancado. Para que los cambios tengan efecto, el administrador debe reiniciar el servicio xinetd. El formato de los archivos en el directorio /etc/xinetd.d/ usan las mismas convenciones que /etc/xinetd.conf. La razón principal por la que la configuración para cada servicio es almacenada en archivos separados es hacer más fácil la personalización y que sea menos probable afectar otros servicios.

Write
Con el comando WRITE se envía algún mensaje a un usuario.
write username [tty]

El mensaje termina con la combinación de teclas CTRL-D.

Talk
TALK es un programa que permite a dos usuarios en el sistema comunicarse escribiendo en el teclado. Al invocar TALK la pantalla se divide en dos partes, cada una correspondiente a uno de los usuarios. Ambos pueden escribir simultáneamente y ambos ven la salida en su parte correspondiente de la pantalla. Para terminar la sesión de TALK, cualquiera de los usuarios puede digitar Ctrl-C. El siguiente comando solicita apertura de una sesión de TALK al usuario1, que debe responder con otro comando similar cuando recibe el pedido.
talk usuario1

El comando MESG permite regular si se desea recibir mensajes o no. Para evitar recibir mensajes de TALK es posible bloquear a otros usuarios el acceso a la terminal donde uno está trabajando; quienes intenten iniciar una sesión recibirán un mensaje indicando que la terminal destino no está habilitada para recibir mensajes. Las opciones son:
Mesg N.- deshabilita recepción de mensajes,
Mesg y habilita recepción de mensajes.
Mesg muestra el estado

Administración de Software

Todo el software en un sistema Linux está dividido en paquetes RPM los cuales pueden ser instalados, actualizados o eliminados. Esta parte describe como manejar los paquetes RPM en un sistema Linux usando herramientas gráficas y de línea de comandos El Administrador

de paquetes (RPM) es un sistema de empaquetado abierto que trabaja en Linux además de otros sistemas Linux y UNIX y que está a la disposición de cualquiera.

RPM's

Package Managment System (Sistema de Administración de Paquetes) es la parte del sistema Linux que permite instalar, desinstalar y gestionar el software de una máquina. El más difundido es el Red Hat package manager (RPM).

RPM facilita las actualizaciones de sistema para el usuario final. Es posible instalar, desinstalar y actualizar paquetes RPM por medio de comandos breves. RPM mantiene una base de datos de los paquetes instalados y de sus archivos y usted puede hacer consultas y verificaciones en su sistema. Si prefiere una interfaz gráfica, puede utilizar herramienta de administración de paquetes para ejecutar muchos comandos RPM. RPM tiene cinco modos de operación básicos (sin contar la construcción de paquetes): instalación, desinstalación, actualización, consulta y verificación.

Rpm [-ivh] [-e] FileName

FileName. El nombre del archivo, como foo-1.0-1.i386.rpm. El nombre de archivo incluye el nombre de paquete (foo), versión (1.0), lanzamiento (1) y arquitectura (i386).

Opciones	Descripción
-iv	Instalación de un paquete
-e	Desinstalación de un paquete
-Uvh	Actualización de un paquete
-q	Consulta. Se tienen las siguientes opciones de especificación con q
-qa	Consulta todos los paquetes actualmente instalados
-qf <file>	Consultará el paquete que posea <file>. Cuando especifique un archivo, deberá especificar la ruta completa del archivo
-qp <packagefile>	Consulta el paquete <packagefile>. Las siguientes opciones son usadas para seleccionar la información que sea de nuestro particular interés Son las llamadas Opciones de Selección de Información
-qi	Presenta información del paquete como nombre, descripción, desarrollo, tamaño, fecha de construcción, fecha de instalación, vendedor y otra información miscelánea
-ql	Presenta la lista de archivos que el paquete "posee"
-qs	Presenta el estado de todos los archivos del paquete. Hay sólo dos posibles estados: normal y perdido
-qd	Presenta un lista de archivos marcados como documentación (paginas "man", paginas "info", readme's, etcétera)
-qc	Presenta una lista de archivos marcados como archivos de configuración. Estos son los archivos que personalizamos después de la instalación para adaptar el paquete a nuestras necesidades (sendmail.cf, passwd, inittab, etcétera)
-V	Verificación, compara la información sobre archivos instalados. Verifica que todos los archivos del paquete estén como cuando fueron originalmente instaladas Entre otras cosas, compara el tamaño, la suma MD5, los permisos, el tipo, el dueño y el grupo de cada archivo. –V se usa con las siguientes opciones
-Vf	Para verificar que un paquete contiene un archivo en particular
–Va	Para verificar todos los paquetes instalados verificar un paquete instalado contra un archivo de paquete RPM
-Vp	Este comando puede ser útil si sospecha que sus bases de datos de RPM están dañadas
-v	Para obtener la lista en el conocido formato ls -l en aquellos opciones que presentan una lista de archivos

En la instalación se imprime una sucesión de gatos a medida que el paquete es instalado, como un medidor del progreso de la instalación.

La instalación de paquetes está diseñada para ser sencilla, pero no exenta de errores. Los errores más comunes a los que nos podemos enfrentar son:

- Ø El paquete que queremos instalar ya este instalado. Esta advertencia protege al sistema, pero si estamos seguros de querer forzar la instalación podemos usar la opción --replacepkgs, lo que le dice a RPM que ignore el error
- Ø Existe un conflicto de archivos. Para hacer que RPM ignore el error, tenemos que usar la opción --replacefiles
- Ø Existen dependencias no resueltas. Este es el lado obscuro del manejo de paquetes de RedHat, para poder instalar este paquete tendríamos que buscar cual es el paquete que

cumple con la dependencia. Si aún queremos instalar el paquete, con la salvedad que no funcione, podemos usar la opción --nodeps. Para resolver este problema usamos el comando YUM, el cual es un manejador de paquetes por encima de RPM que busca en internet los paquetes que sean necesario para cumplir con las dependencias.

En el proceso de desinstalación nos podemos topar de nuevo con el problema de las dependencias. Para hacer que RPM ignore el error y desinstale el paquete de todas maneras (lo cual es una mala idea porque el paquete que depende de éste probablemente falle y no funcione correctamente), utilizamos la opción --nodeps.

En la actualización nos podemos enfrentar al problema de las dependencias, Para obligar a RPM a "actualizar" de todas maneras, podemos usar la opción --oldpackage.

En la opción de verificación, si todo fue verificado correctamente, no habrá salida. Si se encuentran discrepancias, serán mostradas. El formato de la salida es una cadena de ocho caracteres (una c identifica un archivo de configuración) seguido por el nombre del archivo. Cada uno de los ocho caracteres señala el resultado de una comparación entre un atributo del archivo al valor de ese atributo escrito en la base de datos de RPM. Un sólo . (punto) significa que ha pasado la prueba. Los siguientes caracteres señalan que ciertas pruebas no han sido pasadas:

- Ø 5 MD5 suma de verificación
- Ø S tamaño de archivo
- Ø L Enlace simbólico
- Ø T hora de modificación de archivo
- Ø D dispositivo
- Ø U usuario
- Ø G grupo
- Ø ? archivo que no se puede leer

Si ve alguna salida, use su buen juicio para determinar si debería quitar o reinstalar el paquete o resolver el problema de otra manera.

Instalación de software utilizando código fuente

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

- Ø ./configure -prefix=/usr. Este prepara el Makefile y configura las opciones de compilación, mismas que en algunos casos pueden resultar demasiado complejas para un usuario novato. Además verifica si el sistema posee las bibliotecas de desarrollo necesarias para la compilación.
- Ø Make. Este es el que realiza la compilación del código fuente. El proceso puede durar varios minutos.
- Ø make install. Este se encarga de realizar la instalación del los binarios y módulos compilados en los lugares correctos.
- Ø make clean. Opcionalmente podemos utilizar este comando para limpiar los remanentes que se originaron por la compilación a fin de recuperar espacio en el disco duro.
- Ø make uninstall. Si por alguna razón necesita desinstalar el programa resultante, puede utilizar make uninstall.

Yum

(Yellow dog Updater) es una herramienta de administración de paquetes que resuelve de forma automática los problemas de dependencias que nos encontramos al instalar paquetes de forma tradicional con RPM. El único inconveniente es que nuestro equipo tiene que contar con conexión a internet.

yum [opciones] <FileName>

Opciones	Descripción
Update	Para actualizar nuestro sistema
Search	Para realizar una búsqueda
Info	Para obtener información contenida en un paquete
Install	Para instalar un paquete
Remove	Para desinstalar un paquete y todo lo que depende de él
list available	Los paquetes disponibles para instalación y los que tenemos instalados
list Installed	listado de todos los paquetes instalados en el sistema
list updates	Listarán todos los paquetes instalados en el sistema y que pueden (deben) actualizarse

Este comando puede usar con el programa LESS que es una versión mejorada de MORE. Este programa permite visualizar un archivo en forma controlada.

YUM, como resultado de su uso, deja cabeceras y paquetes RPM almacenados en el directorio /var/cache/yum/. Esto ocupa espacio considerable en nuestro preciado disco, por lo que si queremos depurar usamos:

```
yum clean all | Less
```

Creación y Recuperación de Respaldos

Podemos crear y recuperar respaldos utilizando las herramientas GZIP, ZIP, GUNZIP, UNZIP, TAR.

Comprimir con Gzip y Zip

Los archivos comprimidos utilizan menos espacio en el disco y se descargan más rápido que los archivos no comprimidos. Puede comprimir archivos Linux con la herramienta de compresión open-source GZIP o con ZIP, reconocida por la mayoría de sistemas operativos.

Por convención, a los archivos comprimidos se les da la extensión .gz(zip). El comando GZIP (ZIP) crea un archivo comprimido que finaliza con .gz(zip); GUNZIP(UNZIP) extrae los archivos comprimidos y suprime el archivo .gz(zip). Si intercambia archivos con usuarios no-Linux, le conviene zip para evitar problemas de compatibilidad Linux puede abrir archivos zip o gzip fácilmente. Para comprimir un archivo, teclee el siguiente comando en el indicador de comandos de la shell:

```
Gzip(Zip) filename.ext
```

El archivo será comprimido y guardado como filename.ext.gz(zip). Para expandir un archivo comprimido, teclee:

```
Gunzip(Unzip) filename.ext.gz(Zip)
```

El filename.ext.gz(zip) se borra y se reemplaza con filename.ext.

Puede hacer zip o gzip múltiples archivos al mismo tiempo. Haga un listado de los archivos con un espacio entre cada uno.

```
gzip filename.gz file1 file2 file3 /user/work/school
```

El comando anterior comprimirá file1, file2, file3 y el contenido del directorio /user/work/school y los guardará en filename.gz.

Archivar con Tar

Los archivos TAR ubican diferentes archivos o los contenidos de un directorio o directorios en un archivo. Éste es un buen modo de crear copias de seguridad y archivos. Habitualmente, los archivos tar acaban con la extensión .tar.

```
tar –[cvf][xvf][czvf][xzvf] filename.tar file1 file2 filen
```

En este ejemplo, filename.tar representa el archivo que está creando y file1, file2, .. Representan los archivos o directorios que quiere introducir en el nuevo archivo.

Opciones	Descripción
-cvf	Crear un archivo tar
-tvf	Muestra el contenido del archivo tar
-xvf	Extraer los contenidos de un archivo tar Este comando no suprime el archivo .tar file, pero ubica copias del contenido de .tar en el directorio en el que está trabajando actualmente
-czvf	Comprime archivos tar. A los archivos tar se les da generalmente la extensión .tgz y son comprimidos con gzip
-xzvf	Para expandir un archivo tar comprimido

Respaldos

Introducción. En la inmensa mayoría de los sistemas informáticos, los datos almacenados en el sistema tienen mucho mayor costo y son mucho más difíciles de recuperar que el sistema en sí. Entre los riegos de pérdida de datos se cuentan los errores en el software, la falla de equipos, el error humano, el daño intencional, las catástrofes naturales. El respaldo de datos es la generación de una copia, en un momento determinado, de los datos del sistema con vistas a su eventual reposición en caso de pérdida. Todos los sistemas informáticos deben respaldarse cuidadosamente, en momentos predeterminados, siguiendo un cronograma preestablecido.

Dispositivos y Medios

Las causas de pérdida de datos generan los siguientes requisitos de respaldo:

- Uso de un medio removible, utilizable en otra máquina
- Conservación de los respaldos en un lugar físico suficientemente apartado.
- La inmensa mayoría de los dispositivos de respaldo son de tipo magnético. Esto los hace vulnerables a la proximidad de elementos generadores de campos magnéticos. Los respaldos deben mantenerse apartados de estos dispositivos, se aconseja usar medios ópticos o regrabar periódicamente. Puede asumirse una duración de 3 años para los medios magnéticos.
- Muchos fabricantes de unidades de respaldo proveen compresión incorporada.

Características de los principales medios de respaldo

Medio	Capacidad	Reuso	Acceso aleatorio	Comentarios
Disquete	1.44/2.8 MB	Sí	Sí	muy común; poca capacidad, incómodo; poca duración (2 años); útil para respaldo de archivos de configuración o transferencia de archivos chicos; alto costo por MB
Zip	100/250 MB	Sí	Sí	bastante común; varias interfaces de conexión; alto costo por MB
CD-R	650 MB	No	Sí	muy común; varias interfaces de conexión; menor duración que CDs pregrabados, mucho mayor que los medios magnéticos; buenos para datos permanentes, incómodo para respaldos regulares,
CD-RW	650 MB	Sí	Sí	ventajas del CD-R; limitado en capacidad
DVD-R	4.7 a 17 GB	No	Sí	Aún poco común; US$ 5 el de 4.7GB.
Jaz, Orb	2 GB	Sí	Sí	discos removibles; buena velocidad de transferencia
Cinta 8 mm	7 GB	Sí	No	cinta video formato chico; las unidades suelen ser llamadas Exabyte
Cinta 4 mm DAT/DDS	20 GB	Sí	No	DDS (Digital Data Storage) es similar al DAT (Digital Audio Tape) para audio; original 2 GB, DDS-4 en 20 GB; buena velocidad de transferencia; tamaño reducido
Disco fijo	40 GB	Sí	Sí	muy común; excelente transferencia, bajo costo, apto para crear espejos de discos; menor transportabilidad

Existen cintas de nueva tecnología, con mejoras en capacidad, precio, transferencia o duración: Travan (varios fabricantes), ADR (OnStream), DLT (Quantum), AIT (Sony), Mammoth (Exabyte); verificar soporte para el hardware en la versión de UNIX a utilizar. DAT y Exabyte son soluciones baratas para la mayoría de las empresas chicas y medianas; DLT, AIT y Mammoth están orientadas a grandes corporaciones o universidades.

Existen diversos tipos de equipo para cambio automático de volúmenes, de alto costo y con software propio, en capacidades de terabytes. Una adecuada partición en sistemas de archivo, un calendario adecuado y un poco de paciencia permiten respaldar un sistema con razonable esfuerzo.

Los comandos tradicionales de respaldo y recuperación son DUMP y RESTORE. Según los sistemas, pueden tener nombres similares (UFSDUMP, UFSRESTORE en Solaris). Otros comandos también tradicionales son TAR y CPIO, con múltiples opciones de control adaptables a variadas necesidades.

DUMP maneja el sistema de archivos en crudo, leyendo la tabla de inodos para decidir qué archivos deben respaldarse. Esto aumenta su eficiencia, pero obliga a manejar cada sistema de archivos en forma independiente, e impide el respaldo de sistemas de archivos remotos tipo NFS. No obstante, es posible respaldar un sistema de archivos sobre una unidad de respaldo remota usando RDUMP.

El comando DUMP recorre el sistema de archivos haciendo una lista de los archivos modificados o nuevos desde una corrida anterior de DUMP; luego empaqueta todos esos archivos en uno solo y lo vuelca en un dispositivo externo tal como una cinta.

- Soporta multivolumen
- Soporta todo tipo de archivo, incluso de dispositivos;
- Conserva permisos, dueños y fecha de modificación;
- Maneja nombres largos y rutas de anidamiento profundo;

Ø Maneja bien los archivos con huecos (bloques sin datos) producidos por algunos programas; en una copia común estos archivos se agrandan desmesuradamente;
Ø Admite respaldo incremental.

Niveles de un respaldo incremental

El respaldo incremental se implementa asignando un nivel a cada respaldo, de 0 a 9. Un respaldo nivel 0 copia todos los archivos; un respaldo nivel 1 copia sólo los archivos modificados luego de la fecha del último respaldo nivel 0; un respaldo nivel 7 copia sólo los archivos modificados luego del último respaldo nivel 6. La recuperación de un sistema de archivos requiere reponer primero el respaldo nivel 0 y luego sucesivamente el último nivel 1, el último de nivel 2 y siguientes. La información de dump se guarda en el archivo /etc/dumpdates; en caso necesario, este archivo puede ser editado manualmente.

dump Comando Dispositivo Archivo

Opciones	Descripción
0	Nivel de respaldo (completo)
U	Actualiza el registro de respaldo (/etc/dumpdates)
C	Cartucho
F	Arhichivo de respaldo
S	Tamaño de la cinta
D	Densidad de la cinta

dump 0uf /dev/st0 /usr

Crea un respaldo nivel 0 del sistema de archivos /usr usando el dispositivo de cinta con rebobinado /dev/st0 actualizando /etc/dumpdates.

Los dispositivos de cinta suelen tener dos archivos de dispositivo, uno con rebobinado automático (/dev/st0) y otro sin rebobinado (/dev/nst0); ambos se refieren al mismo dispositivo físico. El manejo de la unidad de cinta se hace con el comando MOUNT.

El respaldo en cinta requiere conocer su tamaño y características. El fin de cinta (EOT, End Of Tape) es generalmente detectado, para habilitar los respaldos multivolumen. Un error en el largo de cinta o en la elección de dispositivo rebobinado puede arruinar el respaldo.

dump 5usdf 60000 6250 /dev/st0 /trabajos

Indica un largo de cinta ficticio de 60000 pies, densidad de grabación 6250 cpi. El comando para el manejo de la unidad de cintas es.

mt [-f dispositivo cinta] comando [cantidad]

Los comandos de respaldo en cinta graban una señal de fin de archivo (EOF, End Of File) al terminar de ejecutar; esto permite ubicar un respaldo en particular desplazándose en la cinta. Entre los comandos admitidos por MT se destacan:

Opciones	Descripción
rew	Rebobinado de la cinta
offline	Rebobina la cinta y la expulsa de la unidad
eof	Pone una marca de fin de archivo
status	Da información acerca de la cinta
bsf	Retrocede n bloques
fsf	Avanza n bloques

Esquemas de respaldo

Los niveles de respaldo pueden elegirse arbitrariamente. Un esquema de respaldo se define en función de:

Ø actividad de cada sistema de archivos;
Ø capacidad del dispositivo de respaldo;
Ø redundancia deseada;
Ø cantidad de volúmenes;
Ø complejidad operativa.

Si el sistema de archivos cabe en un volumen, puede hacerse un nivel 0 diario (o semanal), con un grupo de cintas que se va reutilizando; cada N días, la cinta se conserva. Este esquema presenta redundancia masiva y es fácil para restaurar.

Dado que una gran parte de los archivos no cambian, el esquema incremental más simple ya elimina un gran volumen del respaldo diario. La inserción de niveles divide aún más finamente en grupos los archivos activos. El respaldo incremental permite respaldos más frecuentes con menos cintas, más alguna redundancia por repetición de archivos. El esquema de respaldo elegido surge de evaluar estas condiciones.

Recomendaciones generales

Las siguientes recomendaciones no son universales ni infalibles, pero surgen de la experiencia:

- Ø Respaldar todo desde la misma máquina: aunque es más lento, la facilidad de administración y la posibilidad de verificar la corrección del respaldo en todas las máquinas justifica su realización a través de la red. Se corre RDUMP en la máquina remota a respaldar, vía rsh, dirigiendo la salida hacia la unidad de respaldo en la máquina local. Al restaurar, deben tomarse en cuenta eventuales diferencias de sistema operativo; en algunos casos hay inversión de bytes, que pueden arreglarse con DD; esto no resuelve diferencias entre versiones de RDUMP.
- Ø Etiquetar las cintas: fecha, hora, máquina, sistema de archivos, número serial constituyen el mínimo absoluto Los sistemas de archivos / y /usr deben poder restaurarse sin referencia alguna a scripts o información en línea; la sintaxis exacta de los comandos, densidades, opciones y otros valores deben figurar en la documentación de la cinta. Un registro más completo se hace imprescindible cuando se respaldan muchos sistemas de archivos en un mismo volumen.
- Ø Intervalo de respaldos razonable: el sistema de archivos de usuarios puede respaldarse a diario en sistemas grandes o semanalmente en sistemas chicos; la frecuencia de respaldo para otros sistemas de archivos dependerán del uso y la criticidad.
- Ø Respaldos diarios en un solo volumen: buscar un esquema o un medio para que el respaldo diario quepa en un solo volumen; es la única forma simple de realizar un respaldo diario, cargando la cinta a última hora y ejecutando el respaldo en CRON. Recordar: el respaldo de varios sistemas de archivos en una cinta única requiere usar el dispositivo no rebobinado y documentar bien las cintas.
- Ø Crear sistemas de archivo acordes con el tamaño del medio de respaldo: con las capacidades actuales, es posible crear sistemas de archivo de tamaño razonable que quepan en una cinta. Debe haber una buena razón para crear sistemas de archivo muy grandes.
- Ø Mantener las cintas fuera del lugar: pero realmente en otro lugar, apartado del lugar de la instalación; hay un sin número de historias sobre pérdida de los respaldos conjuntamente con el sistema.
- Ø Seguridad del respaldo: el robo de un respaldo equivale al robo de toda la información vital de una organización. Las precauciones de seguridad con los respaldos debe ser tanta como la dispensada al propio sistema, con el agravante de que es más fácil llevarse unas cintas que la información en disco.
- Ø Limitar la actividad durante dump: la actividad en los archivos mientras se ejecuta dump puede ocasionar confusión en el momento de restaurar. Esto no es tan crítico en niveles superiores, pero sí en el nivel 0. Puede hacerse el respaldo en horas de escasa actividad, nocturnas o en fin de semana. Es preciso cuidar de no coincidir con los programas del sistema que modifican el sistema de archivos; éste debe permanecer estacionario mientras se realiza el respaldo. La solución más segura es hacer el respaldo en monousuario. En ocasiones especiales, como al encarar una actualización del sistema operativo, deberá uno armarse de paciencia, bajar la máquina a monousuario y respaldar el sistema en nivel 0. Existen programas especiales (archivadores o ”filers”) capaces de tomar un registro periódico del estado del sistema y resincronizar el respaldo. Debe considerarse esta posibilidad cuando no sea posible reducir la actividad del sistema en ningún momento.
- Ø Verificar el respaldo: hay también historias de respaldos tomados cuidadosamente que nunca pudieron restaurarse. Releer la cinta con RESTORE para generar la tabla de contenido es una prueba razonable; probar recuperar un archivo en particular obliga a recorrer partes más alejadas de la cinta, ofreciendo una comprobación más sólida.
- Ø Vida útil de las cintas: como todo en el mundo, las cintas tienen una vida limitada, indicada por los fabricantes en cantidad de pasadas. Un respaldo, una restauración o un salto de archivos representan, cada uno, una pasada.

Recuperación

Restaurar archivos

1) Determinar en qué cinta se encuentran los archivos a restaurar. Los usuarios generalmente buscan la última versión del archivo, pero no siempre es así. La existencia y ubicación del archivo dependen del esquema de respaldo empleado. Si se conservan catálogos (listas con todos los archivos respaldados en una fecha), la búsqueda se simplifica: basta con verificar si el archivo buscado se encuentra en el catálogo. Si no es así, se deberán revisar las cintas más probables según la fecha indicada por el usuario o recorrer todo el conjunto desde el respaldo nivel 0 inmediato anterior.
2) Crear un directorio donde recuperar los archivos. Muchas versiones de restore requieren reponer la ruta entera de directorios para recuperar el archivo. No usar /tmp; su contenido será borrado en un rearranque imprevisto.

3) Si se han colocado varios archivos de respaldo en una misma cinta, se deberá consultar la documentación de ubicación de cada uno, determinar el lugar en que se encuentra el de interés y usar el comando MT para ubicar el comienzo del archivo de respaldo.
4) Restaurar el archivo. Usar el comando complementario del respaldo: si se usó DUMP para respaldar, usar RESTORE; si se usó RDUMP, usar: RESTORE.
5) Entregar el archivo al usuario. Se puede copiar el archivo hacia el directorio del usuario, verificando que no exista ya un archivo con ese nombre. En ningún caso debe sobrescribirse un archivo de otro usuario. Otra alternativa es dejarlo en el lugar de recuperación para que el usuario lo copie. En este caso, será preciso limpiar regularmente el directorio de recuperación.

RESTORE admite la opción i, para uso interactivo: el comando lee el catálogo de la cinta; se recorren los archivos como si se tratara de un árbol de directorios común, usando LS, CD y PWD; se van seleccionando los archivos a restaurar con ADD; cuando se han seleccionado todos, indicando EXTRACT se los recupera de la cinta.

```
restore if roble:/dev/nrst1
restore> add perdido.arch
```

El archivo se agrega a la lista de archivos a recuperar. Agregar un directorio agrega todo su contenido. El asterisco indica que está marcado para recuperar.

```
restore > extract
```

Muestra mensajes; si no se sabe en qué volumen está el archivo, debe comenzarse por el último y proceder hacia el principio; aquí asumimos saber que está en el primer volumen.

Specify next volume #:.1

Se realiza la extracción; pregunta si el directorio raíz de la cinta debe interpretarse como directorio corriente; se usa sólo al restaurar sistemas de archivo completos.

set owner mode for '.'? [yn] n

Restaurar sistemas de archivos

Antes de restaurar un sistema de archivos completo, se debe estar seguro de haber eliminado las causas que provocaron su destrucción.

1) Crear un sistema de archivos en la partición donde se va a restaurar; crearlo usando el comando NEWFS dispositivo nombre_estructura.
2) Montar el nuevo sistema de archivos con MOUNT
3) Cambiar al directorio raíz (punto de montaje) del nuevo sistema de archivos. Montar la primera cinta del último respaldo nivel 0. Arrancar la restauración con RESTORE . El comando pedirá las cintas sucesivas.
4) Al terminar de restaurar el nivel 0, continuar con los diferentes niveles en el mismo orden del esquema de respaldos empleado.

Esta secuencia repone el sistema de archivos a su estado original más cercano al momento de pérdida. En el esquema de respaldos empleado, la única diferencia es la mágica aparición de los archivos que fueron borrados. Hay versiones de restore que llevan registro de los archivos borrados.

En una actualización del sistema operativo, debe hacerse un respaldo nivel 0 antes de la actualización, efectuar luego la actualización cuidando de reponer los archivos de configuración necesarios en los sistemas de archivos afectados por la actualización. Una vez que todo esté funcionando, realizar inmediatamente un nuevo respaldo nivel 0. Esto es imprescindible para asegurar la coherencia de los siguientes niveles ya que la actualización puede haber modificado fechas de archivos preexistentes.

El comando que se usa para recuperar archivos en solaris se llama UFSRESTORE

Otros comandos de respaldo

Los clásicos comandos TAR (BSD) y CPIO (System V) sirven para respaldar archivos, directorios y grupos diversos de archivos y directorios. Se usan frecuentemente para trasladar información, distribuir software o aún copiar árboles de directorios dentro de un mismo sistema, sobre todo por su capacidad de conservar dueños, permisos y fechas, no siempre posible con CP.

Copia de un árbol de directorios usando TAR:

```
tar cf dir origen | ( cd dir destino ; tar xfp - )
```

Copia de un árbol de directorios usando CPIO:

```
find dir origen -depth -print | cpio -pdm dir destino
```

Este comando es poco usado, habiendo cedido lugar a TAR.

Al usar estos comandos verificar estas posibles limitaciones: soporte multivolumen, nombres de ruta largos, recuperación de errores, fijación de factor de bloqueo (20 por defecto). Muchas han sido levantadas, en particular en las versiones de GNU, pero conviene verificar en la documentación y comprobar en la práctica.

Cuando es preciso realizar transformaciones de datos es útil el comando DD (data duplicator). Sin parámetros, este comando sólo copia entrada en salida. Admite un cierto número de opciones que permiten cambiar formato de los datos o transferir entre distintos medios.

Copia de una cinta entre dos unidades:

```
dd if=/dev/rmt8 of =/dev/rmt9 cbs=16b
```

Copia de una cinta en una sola unidad:

```
dd if=/dev/rmt8 of =/tmp/cinta.archdev cbs=16b
```

cambiar la cinta;

```
dd if=/tmp/cinta.arch of =/dev/rmt8 cbs=16b
```

Conversión desde cinta con diferente orden de byte:

```
dd if=/dev/rst8 conv=swab | tar xf
```

VOLCOPY permite realizar una copia exacta de un sistema de archivos completo hacia otro dispositivo, eventualmente con cambio en el bloqueo.

Amanda (Advanced Maryland Automatic Network Disk Archive) es un software para respaldo de red capaz de actuar en una LAN hacia una unidad de cinta ubicada en un servidor, cumpliendo todas las exigencias deseables para un administrador. Usa DUMP y RESTORE como comandos de base, pero también puede manejar TAR de GNU y SMBTAR de Samba para respaldar datos de máquinas Windows. Corre en muchas versiones de UNIX, soporta extensa variedad de hardware, permite compresión sobre el cliente antes de la transferencia, graba registros completos del respaldo en la cinta. Amanda es software libre, escala bien, es muy configurable, evoluciona rápido; está siendo usado extensamente en todo el mundo.

Monitoreo del Sistema

Podemos 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. Los siguientes comandos nos permiten ver las tareas que se están ejecutando en nuestro sistema y la cantidad de recursos que ocupa cada uno de ellos

Monitoreo de procesos.

El siguiente comando sirve para listar procesos:

```
ps opciones
```

Opciones	Descripción
-f	Presente la información en formato completo
-e	Presenta cada uno de lo procesos
-a	Presenta todos los procesos en una terminal
-u	Muestra los controladores

TOP Muestra información de los procesos dentro del sistema que más demanda de recursos tienen. Así como porcentajes de memoria y procesador están siendo utilizados en tiempo real.

Espacio en disco.

El siguiente comando indica cuando espacio estamos utilizando en todas las particiones.

```
df opciones dispositivo
```

Opciones	Descripción
-h	Describe la información indicando sus unidades
-k	Expresa las cantidades en kilobytes

```
S.ficheros Bloques de 1K    Usado  Dispon Uso% Montado en
/dev/hda3        15116868 14212968  135996 100%         /
    none           257496        0  257496   0%   /dev/shm
/dev/hda2        10238812  7572620 2146088  78%  /mnt/hda2
/dev/hda6        25466592 25197552  269040  99%  /mnt/hda6
```

La primera columna nos indica el filesystem o sistema de archivos o dispositivo físico o en términos más comprensibles, el disco. La segunda nos dice el espacio total en el disco en unidades de 1024 bytes o 1 kilobyte, aunque las unidades varían de una implementación a otra. La tercera y cuarta columnas indican el espacio utilizado y el espacio disponible en las mismas unidades que la segunda. La quinta columna indica el porcentaje ocupado del disco. Por último, la sexta columna indica en que parte del sistema de archivos está montado el disco. Cabe hacer la aclaración que se provee de un mecanismo para acotar el espacio en disco disponible para cada usuario, pero la mayoría de las veces esta restricción no se aplica, con lo cual si hay cinco usuarios en el sistema, estos cinco usuarios compiten por el espacio disponible en el disco donde están montados sus directorios. La elección de restringir el espacio en disco a cada usuario está determinada por las políticas de uso que emplee el administrador del sistema.

El siguiente comando sirve para determinar cuando espacio tiene un archivo o directorio.

du opciones archivo

Opciones	Descripción
-sh	Presenta la suma con abreviatura de las unidades
-sk	Presenta la suma en kilobytes

```
9821824 .
7051356 ./vmware
5252336 ./vmware/Windows XP Professional x64 Edition
2209312 ./Centos4.3
1799016 ./vmware/Red Hat Enterprise Linux 4 2
169820 ./.evolution
```

Monitoreo de la red

- Ø IPCONFIG permite configurar las interfaces de red y mostrar información sobre ellas.
- Ø NMAP Realiza un mapeo de puertos a cualquier servidor, para determinar que sistema operativo esta utilizando o que puertos tiene abiertos. Su sintaxis es nmap –p 80-1024 servidor
- Ø PING Permite enviar paquetes de información a través de la red para determinar si se tiene acceso a una maquina remota.
- Ø TRACEROUTE Permite trazar una ruta de hosts por donde pasa un paquete para llegar al host destino seleccionado. Su sintaxis es TRACEROUTE host

Bitácoras del sistema

Como administrador de una máquina Linux, revisa los archivos de registro. Sirven para detectar posibles fallos de seguridad en nuestra máquina Linux. Los archivos de registro se encuentran en /var/log:

- Ø messages: mensajes de seguridad y autenticación.
- Ø cron: mensajes generados por el demonio cron.
- Ø secure: mensajes de seguridad de acceso (conexiones, wrappers...).
- Ø access_.log: mensajes de acceso al servidor WWW, si lo hubiese.
- Ø Error_log: mensajes de error durante el acceso al servidor WWW, si lo hubiese.
- Ø utmp: contiene información sobre los usuarios que se encuentran actualmente en el sistema. La información la consultamos con comandos como who,w,users...
- Ø wtmp: contiene información sobre las entradas y salidas de los usuarios al sistema. La información la consultamos con el comando LAST.

Si ves que en estos archivos se repiten registros continuos desde máquinas desconocidas, probablemente tu máquina esté siendo objeto de una posible incidencia de seguridad.

El archivo de configuración de los registros es el /etc/syslog.conf y el demonio encargado de registrar los eventos es syslogd.

Monitoreo de recursos

Antes de poder monitorear recursos, primero tenemos que conocer cuales recursos hay que supervisar. Todos los sistemas tienen disponibles los siguientes recursos:

- Ø CPU
- Ø Memoria
- Ø Almacenamiento
- Ø Ancho de banda

Las herramientas disponibles en CentOS para monitorear estos recursos son: FREE, TOP, VMSTAT, SYSSTAT, MPSTAT.

FREE muestra la utilización de la memoria del sistema.

$ free

```
                 total       used       free     shared    buffers     cached
Mem:            333176     210096     123080          0      18292     146648
-/+ buffers/cache:          45156     288020
Swap:           688120          0     688120
```

La columna Mem: muestra la utilización de la memoria física, mientras que la columna Swap: muestra la utilización del espacio de intercambio (swap) del sistema. La columna -/+ buffers/cache: muestra la cantidad de memoria actualmente dedicada a las memorias intermedias del sistema (buffers).

Esta herramienta nos es útil para determinar si un problema relacionado con la memoria está en progreso actualmente. Aunque FREE tiene la opción de mostrar repetidamente los números de utilización de memoria a través de su opción -s, la salida se desplaza, haciendo difícil detectar cambios en la utilización de memoria.

TOP es una herramienta completa que nos monitorea: utilización del CPU, estadísticas de procesos, utilización de memoria es decir todo. Además, a diferencia de FREE command, el comportamiento predeterminado de TOP es el de ejecutarse de forma continua.

```
Tasks: 57 total, 1 running, 56 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.4% us, 1.5% sy, 0.1% ni, 97.7% id, 0.4% wa, 0.0% hi, 0.0% si
Mem: 333176k total, 210416k used, 122760k free, 18460k buffers
Swap: 688120k total, 0k used, 688120k free, 146696k cached

PID USER PR  NI VIRT RES SHR S %CPU %MEM TIME+   COMMAND
  1 root 16   0 2580 560 480 S  0.0  0.2 0:01.41 init
  2 root 34  19    0   0   0 S  0.0  0.0 0:00.00 ksoftirqd/0
  3 root  5 -10    0   0   0 S  0.0  0.0 0:01.53 events/0
  4 root  5 -10    0   0   0 S  0.0  0.0 0:00.02 khelper
```

La pantalla se divide en dos secciones. La parte superior contiene información relacionada con el estatus general del sistema: tiempo ejecutándose, carga promedio, cuentas de procesos, estado del CPU y estadísticas de utilización para la memoria y el espacio de intercambio. La sección de abajo muestra estadísticas a nivel de procesos. Es posible cambiar lo que se muestra mientras TOP se ejecuta. Por ejemplo, por defecto TOP muestra procesos activos y ociosos. Para mostrar solamente procesos activos o que no estén ociosos, presione [i]; otro toque lo retorna al modo de visualización predeterminado.

Con el comando VMSTAT, es posible obtener una vista general de los procesos, memoria, swap, E/S, sistema y actividad de CPU:

```
$ vmstat
procs memory                    swap  io       system cpu
r b   swpd   free  buff  cache si so bi bo   in cs  us sy id wa
0 0      0 122440 18832 146720  0  0  9  4 1003 42   0  1 98  0
```

La primera línea divide los campos en seis categorías, incluyendo: procesos, memoria, swap, E/S, sistema y estadísticas relacionadas al CPU. La segunda línea identifica aún más los contenidos de cada campo, de acuerdo a lo siguiente:

- Los campos relacionados a los procesos son:
  - r El número de procesos ejecutables esperando para acceder al CPU
  - b El número de procesos en un estado dormido continuo
- Los campos relacionados a la memoria son:
  - swpd La cantidad de memoria utilizada
  - free La cantidad de memoria libre
  - buff La cantidad de memoria utilizada por las memorias intermedias
  - cache La cantidad de memoria utilizada como caché de páginas
- Los campos relacionados a swap son:
  - si La cantidad de memoria intercambiada desde el disco
  - so La cantidad de memoria intercambiada hacia el disco
- Los campos relacionados con E/S son:
  - bi Los bloques enviados a un dispositivo de bloques
  - bo Los bloques recibidos desde un dispositivo de bloques
- Los campos relacionados al sistema son:
  - in El número de interrupciones por segundo
  - cs El número de cambios de contexto por segundo
- Los campos relacionados al CPU son:
  - us El porcentaje de tiempo que el CPU ejecutó código de nivel del usuario
  - sy El porcentaje de tiempo que el CPU ejecutó código de nivel del sistema
  - id El porcentaje de tiempo que el CPU estaba desocupado
  - wa Esperas de E/S

SYSSTAT nos sirve para llevar un histórico del desempeño del equipo. SYSSTAT contiene las siguientes herramientas relacionadas con reunir estadísticas de E/S y CPU:

- IOSTAT. Muestra una descripción general de la utilización del CPU, junto con las estadísticas de E/S para uno o más unidades de disco.
- MPSTAT. Muestra estadísticas de CPU más complejas.

SYSSTAT también contiene herramientas para reunir datos de utilización de los recursos y crear informes diarios basados en esos datos. Estas herramientas son:

- SADC. Conocida como el coleccionador de datos de actividad del sistema. Reúne información sobre la utilización de recursos y la escribe a un archivo.
- SAR. Los informes SAR, producidos a partir de los archivos creados por SADC, se pueden generar interactivamente o se pueden escribir a un archivo para un análisis más intensivo.

Por default esta suite de herramientas no viene instalada. Para instalarla podemos usar el siguiente comando:

yum install sysstat

El comando IOSTAT en su forma más básica proporciona una descripción general de las estadísticas del CPU y E/S de disco:

iostat

```
Linux 2.6.9-34.EL (centos) 13/07/06
cpu-med: %user %nice %sys %iowait %idle
          0.44  0.05 1.43    0.40 97.68

Device:  tps Blq_leid/s Blq_escr/s Blq_leid Blq_escr
hdc     0.00       0.05       0.00       848         0
sda     0.73      17.91       8.72    315063    153356
sda1    0.03       0.05       0.00       944         4
sda2    1.45      17.80       8.72    313151    153352
dm-0    1.43      17.76       8.72    312402    153352
```

Debajo de la primera línea (la cual contiene la versión del kernel del sistema y el nombre del host, junto con la fecha actual), IOSTAT muestra una vista general de la utilización promedio del CPU desde el último arranque. El informe de utilización del CPU incluye los porcentajes siguientes:

- Porcentaje de tiempo en modo usuario (ejecutando aplicaciones, etc.)
- Porcentaje de tiempo en modo usuario (para procesos que han alterado su prioridad de planificación usando NICE)
- Porcentaje d tiempo usado por el sistema
- Porcentaje de tiempo en modo kernel
- Porcentaje de tiempo ocioso

Debajo del informe de utilización del CPU está el informe de utilización de dispositivos. Este informe contiene una línea para cada dispositivo en el sistema e incluye la información siguiente:

- El número de transferencias (u operaciones de E/S) por segundo.
- El número de bloques de 512 bytes leídos por segundo.
- El número de bloques de 512 bytes escritos por segundo.
- El número total de bloques de 512 bytes leídos.
- El número total de bloques de 512 bytes escritos.

El comando MPSTAT aparece primero sin diferencias con el informe de utilización de CPU producido por IOSTAT:

mpstat

```
Linux 2.6.9-34.EL (centos) 14/07/06
00:06:14 CPU %user %nice %system %iowait %irq %soft %idle  intr/s
00:06:14 all  0.38  0.04    1.28    0.35 0.01  0.00 97.94 1002.83
```

Con la excepción de una columna adicional mostrando las interrupciones por segundo manejadas por el CPU, no hay diferencia real. Sin embargo, la situación cambia si se utiliza la opción de MPSTAT, -P ALL.

mpstat -P ALL

```
Linux 2.6.9-34.EL (centos) 14/07/06
00:06:08 CPU %user %nice %ystem %iowait %irq %soft %idle  intr/s
00:06:08 all  0.38  0.04   1.28    0.35 0.01  0.00 97.94 1002.83
00:06:08   0  0.38  0.04   1.28    0.35 0.01  0.00 97.94 1002.83
```

En sistemas multiproceso, MPSTAT permite desplegar de forma individual la utilización de cada CPU, haciendo posible determinar que tan efectivamente se utiliza cada CPU.

El Intérprete de Comandos de Unix, Shell

Fundamentos

El shell es un programa interpretador de comandos que lee cada comando que ingresa el usuario y dispone lo necesario para que se ejecuten. Bash, "Bourne Again SHell", fue creado para usarlo en el proyecto GNU. La intención fue que fuese el intérprete de comandos estándar en el sistema GNU. “Nació” oficialmente el domingo, 10 de enero de 1988. Brian Fox fue quien programó las primeras versiones de Bash y continuó actualizándolo hasta 1993. A principios de 1989, Chet Ramey empezó a ayudar a Brian y fue el responsable de muchos arreglos en el código y nuevas características. Ahora el mantiene oficialmente el shell bash siendo la última versión la 2.x. Es también un lenguaje de programación. Los programas escritos con el shell se llaman scripts. Recuerda que UNIX diferencia entre mayúsculas y minúsculas.

Para poder iniciar sesión remota en un sistema Unix, necesitamos un emulador de terminal.
putty
secure shell client

Para entrar a sesión necesitamos un identificador de usuario, conocido como login y una contraseña. El password o contraseña no es desplegado por seguridad, si los datos son correctos entraremos a sesión.

El símbolo de $ es el llamado prompt y nos indica que el shell esta esperando por nuestras órdenes.

Las teclas Ctrl+D también terminan la sesión.

El intérprete de comandos o “shell” actúa entre el sistema operativo y el operador. Provee al usuario una interfaz hacia el sistema operativo. El usuario dialoga con el intérprete de comandos y éste transfiere las órdenes al sistema operativo, que las ejecuta sobre la máquina. En dicha orden, puede haber programas internos o externos: Los programas internos son aquellos que vienen incorporados en el propio intérprete, mientras que los externos son programas separados (aplicaciones de /bin,/usr/bin,. . .)

En el mundo Linux/Unix existen tres grandes familias de shell´s. Estas se diferencian entre sí básicamente en la sintaxis de sus comandos y en la interacción con el usuario.

Interpretes de comandos en Linux/Unix

Tipo de Shell	Shell estándar	Clones libres
AT&T Bourne shell	Sh	ash, bash, bash2
Berkeley “C” shell	csh	Tcsh
AT&T Korn shell	ksh	pdksh, zsh
Otros interpretes	—	esh, gush, nwsh

La estructura general de un comando es la siguiente:
nombre opciones argumentos

nombre es el nombre del comando. Las opciones o banderas controlan la forma en que actúa el comando; van precedidas por el signo - (menos). Los argumentos son comúnmente nombres de archivos o nombres de login de usuarios.

Los comodines son caracteres que sustituyen cadenas de caracteres.

Caracter	Descripción
*	Secuencia de caracteres cualesquiera, 0 o más.
?	Carácter cualquiera, uno y uno sólo; debe aparear un carácter.
[: : :]	Sustituye los caracteres indicados individualmente.

Variables de ambiente o entorno

Una variable es el nombre que nos refiere a un área de almacenamiento temporal en la memoria. Las variables contienen información usada para personalizar el shell e información requerida para que otros procesos funcionen de manera adecuada. El shell permite que almacenemos valores en variables. El shell permite definir variables que son exportadas a subprocesos y variables que no.
Nombre = valor

Nombre es cualquier cadena de caracteres que no incluya $ ni espacio; valor es cualquier cadena; puede incluir el espacio si el valor está entre comillas. Para exhibir el contenido de una variable de ambiente se escribe ECHO $nombre-variable.

Es una costumbre muy arraigada en UNIX usar mayúsculas para los nombres de variable, así como es una regla usar minúsculas para los comandos. Las opciones pueden ser mayúsculas y minúsculas; la opción -a no es lo mismo que -A.

Las variables de ambiente pueden ser usadas como nombres de comando o como argumentos de un comando, algunas son útiles para no tener que escribir muchas opciones al ejecutar un programa, otras las utiliza el propio shell (PATH, PS1,. . .).

ENV muestra las variables de ambiente definidas. Muchas son fijadas en el ingreso del usuario al sistema (variables de login); y otras son propias del shell (variables del shell).

Variables de Entorno más usuales

Variable	Descripción
DISPLAY	Donde aparecen la salidas de X-Windows
HOME	Directorio personal
LOGNAME	muestra el nombre de login del usuario
HOSTNAME	Nombre de la máquina
MAIL	Archivo de correo
PATH	Especifica la lista de directorios separados por dos puntos (:), en donde se buscaran los comandos que se desean ejecutar.
PS1	Prompt
PS2	Especifica el prompt secundario del shell
SHELL	Intérprete de comandos por defecto
TERM	Tipo de terminal
USER	Nombre del usuario
NOEXISTE	Las variables no definidas no muestran nada cuando son referenciadas

La forma de definir una variable de entorno cambia con el intérprete de comandos, se muestra tcsh y bash siendo los dos más populares en el ámbito Linux:
Ø bash: export VARIABLE=Valor
Ø tcsh: setenv VARIABLE Valor

Resumen de comandos que afectan variables

Acción	Comando
Para configurar una variable	VARIABLE=valor
Para eliminar una variable	unset VARIABLE
Para desplegar todas las variables	set, env o export
Para desplegar el valor almacenado en una variable	echo $variable

Entrecomillado de argumentos

Los espacios separan argumentos. El entrecomillado obliga a tratar una cadena con espacios como si fuera un solo argumento. Cuando se usan comillas dobles (" ") el shell interpreta las variables de ambiente incluidas, actuando según su contenido.

Cuando se usan comillas simples (' ') el shell no interpreta las variables de ambiente, tratando sus nombres como cadenas. Las comillas simples permiten usar comillas dobles.

El acento grave (`) asigna a una variable de ambiente la salida estándar de un comando. Es decir, permite ejecutar un comando dentro de otro, encerrando el comando entre acentos graves. El siguiente procedimiento se usa para escribir el nombre de login del usuario:

```
MUESTRA=`echo $LOGNAME`
echo $MUESTRA
$ afernandez
```

PS1

La variable símbolo de indicador de comandos nivel 1 PS1 (Prompt Symbol level 1) es un valor de ambiente que puede no aparecer en la salida de ENV.

```
   echo $PS1
$
```

responde con el indicador de comandos actual, $.

```
   PS1=hola:
hola:
```

Entrada estándar y salida estándar.

Los comandos leen como entrada una secuencia de caracteres (flujo de entrada o ”input stream”) y escriben a la salida otra secuencia de caracteres (flujo de salida o ”output stream”). Estas secuencias no tienen estructura interna alguna y se les llama entrada estándar y salida estándar.

```
cat nota > nota2
```

> redirige la salida estándar a un archivo que crea o sobrescribe, en vez de a la pantalla.

```
mail juan <nota
```

< toma la entrada estándar de un archivo en vez del teclado.

Los comandos y los archivos tienen una estructura orientada a carácter. En muchos comandos se puede intercambiar teclado por archivos o dispositivos de hardware.

```
cat nota
cat <nota
```

son comandos equivalentes porque cat opera sobre un archivo y también sobre la entrada estándar.

```
cat nota LEAME nueva.nota
cat nota - nueva.nota <LEAME
```

Son comandos equivalentes. El signo menos (-) aislado equivale a tomar en ese punto la entrada estándar, que usualmente está asignada al teclado.

Fin de flujo.

El ingreso de datos desde el teclado, así como el despliegue en pantalla, se manejan en UNIX como ”flujos de caracteres”, una serie de caracteres uno tras otro. Para indicar en el teclado el fin de un flujo de caracteres se usa Ctrl-D. Este símbolo no forma parte del ingreso; simplemente indica el final del ingreso, que ya no se escribirá más. En UNIX no hay un carácter especial para indicar fin de archivo; el sistema sabe cuando termina un archivo por medio de contadores.

```
cat nota - nueva.nota > arch.salida
```

Procesa el archivo nota, lee lo que se digite en el teclado hasta recibir un Ctrl-D, procesa nueva.nota y escribe todo en el archivo arch.salida; se reúnen tres fuentes distintas en una única salida.

El símbolo >> redirige la salida estándar a un archivo, pero agregando al final del mismo en lugar de reemplazar su contenido.

Error estándar.

Además de los flujos de entrada y salida estándar, existe un tercer flujo de caracteres, el error estándar, hacia donde se dirigen los mensajes de error. El error estándar está dirigido habitualmente a la pantalla, pero mediante 2> o 2>> se redirige hacia un archivo de errores. Los flujos estándar se reconocen por los siguientes números:

- Ø 0 la entrada estándar, usualmente el teclado;
- Ø 1 la salida estándar, usualmente la pantalla;
- Ø 2 el error estándar, usualmente la pantalla.

```
echo Archivo de errores > salida.error
cat noexiste 2>> salida.error
cat salida.error
```

El archivo salida.error contiene el mensaje inicial y el de error.

```
ls noexiste 2>>salida.error
rm noexiste 2>>salida.error
cat salida.error
```

Reúne los mensajes de error en el archivo salida.error.

Interconexión de comandos (entubamiento).

El operador | (“pipe”) hace que la salida del comando precedente sea la entrada del comando siguiente, creando un entubamiento o interconexión de comandos.

```
cat nota LEAME | pr >salida
```

Hace que la concatenación de los archivos nota y LEAME sea servida al comando pr, cuya salida está redirigida a un archivo.

Filtros.

Muchos comandos están pensados para ser interconectados, pasando simplemente la entrada hacia la salida, por lo que se les llama habitualmente filtros.

Filtros	Función
Sort	Ordena las líneas de un texto
Cut –fN1, N2 –dD	Corta y presenta secciones de una línea. Donde f indica que se refiere a los campos identificados por N1, N2 y delimitados por D
Cut –cI-F	Corta y presenta secciones de una línea. Donde c indica que son columnas y I el identificador de la inicial y F el de la final. En caso de omisión de alguna se presenta desde el principio o final
Od	Convierte archivos a forma octal u otras
Paste	Une líneas de diferentes archivos
Tac	Concatena e imprime archivos invertidos
Tr V1N1 V2N2	Traduce o borra caracteres, Cambiando toda ocurrencia de V1 por N1 y de V2 por N2
Uniq	Remueve líneas repetidas
Wc	Cuenta según la opción -l para líneas, -w para palabras y -c para caracteres.

Algunos filtros han llegado a ser tan complejos que son en si, un lenguaje de procesamiento de texto, de búsqueda de patrones, de construcción de scripts y muchas otras posibilidades. Entre ellos podemos mencionar herramientas tradicionales en Linux/Unix como AWK y SED y otras más modernas como PERL.

Campos y delimitadores.
Un campo es una cadena de caracteres separada por un carácter delimitador. El archivo /etc/passwd tiene en cada línea una serie de campos separados por dos puntos (:) .
```
cut -f1 -d: </etc/passwd
cut -f1,3,5 -d: </etc/passwd
```

Valores de retorno de los comandos.
Los comandos devuelven un código de retorno 0 si el comando termina correctamente o un número entre 1 y 255 según la razón de falla. El código de retorno del último comando queda en una variable llamada '?', que se interroga como $?. $? es una 'variable de shell' mantenida internamente por el propio intérprete. Otras variables de shell son:
Ø # número de argumentos en el comando para la shell actual
Ø $ número de proceso para el shell actual

Estas variables se interrogan como $# y $$.

Secuencias de comandos.
El shell es también un lenguaje de programación. Pueden escribirse varios comandos en una misma línea separándolos con; (punto y coma).
```
date ; echo Hola ; echo $LOGNAME MUESTRA=`date; echo Hola ; echo LOGNAME`; echo MUESTRA
```

Redirección del shell.
El shell que atiende los comandos del usuario es el login shell; arranca cuando el usuario ingresa al sistema y termina cuando sale. Escribir sh como comando invoca una segunda instancia del shell, que puede terminarse con el comando exit o con Ctrl-D.

Comentarios (de # en adelante)

Para convertir el archivo en ejecutable para el usuario, hacer
```
chmod u+x datos.usuario
```

Comandos en background
Linux, como cualquier sistema Unix, puede ejecutar varias tareas al mismo tiempo. En sistemas monoprocesador, se asigna un determinado tiempo a cada tarea de manera que al usuario le parece que se ejecutan al mismo tiempo.

Para ejecutar un programa en background, basta con poner el signo ampersand (&) al término de la línea de comandos. Cuando ha terminado la ejecución del programa, el sistema lo reporta mediante un mensaje.

Si se hubiese ejecutado el programa y no se hubiese puesto el ampersand, se podría pasarlo a background de la siguiente manera:
1) Se suspende la ejecución del programa, pulsando Ctrl+Z.
2) Se ejecutamos la siguiente orden: BG

Alias

Un “alias” es un nombre alternativo para un comando. Así, en lugar de escribir el comando propiamente dicho, escribiríamos el alias de dicho comando. Un alias se puede definir por varios motivos, por ejemplo:

- Ø Dar nombres familiares a comandos comunes:
  - Ø alias md=’mkdir’
- Ø Dar nombres a comandos largos:
  - Ø alias tbz2=’tar -cv --use-compress-program=bzip2 -f’

Para no tener que escribir todos los alias siempre que entremos al sistema, escribiríamos dicho alias en el archivo /.bash profile.

Reutilización de comandos

El shell almacena una historia de los comandos que el usuario ha escrito. Por medio de esta historia es posible volver a ejecutar una orden que ya se ha escrito anteriormente sin tener que escribirla de nuevo.

El comando HISTORY muestra la secuencia de comandos, con un número a su izquierda. Con este número es posible llamar de nuevo el comando utilizando el carácter admiración “!”; Por ejemplo history retorna

1) history
2) ls
3) cd public_html
4) ls
5) rm *.bak
6) history

Y para ejecutar nuevamente el comando rm *.bak solo es necesario escribir !5. También se puede pedir el último “rm” que se ha ejecutado escribiendo !rm.

El último comando se repite con doble admiración “!!”. Es posible también editar el último comando utilizando el carácter “^” pero este conocimiento se esta volviendo poco útil ya que los nuevos shells permiten viajar por la “historia” y editar los comandos usando únicamente las flechas del teclado.

Archivos de bash. Cada shell posee ciertos archivos donde mantiene su configuración. Estos tienen una jerarquía que va desde el archivo general de configuración del sistema para todos los shells, pasando por el archivo propio del shell, hasta los archivos personales del usuario.

Archivos de bash

Archivo	Descripción
/bin/bash	Ejecutable bash
/etc/profile	Archivo de inicialización utilizado por los shells
~ /.bash profile	Archivo(s) de inicialización personal
~ /.profile	utilizado por los shells
~ /.bash login	Ejecuta cuando entra al shell
~ /.bash logout	Ejecuta cuando sale del shell
~ /.bashrc	Archivo personal de inicialización del shell
~ /.inputrc	Archivo de inicialización individual

Resumen de comandos básicos en Unix

Comando/Sintaxis	Descripción	Ejemplos
chown usuario:grupo fich	Cambia el dueño un archivo	chown nobody miscript
cp fich1. . . fichN dir	Copia archivos	cp foo foo.backup
diff [-e]arch1 arch2	Encuentra diferencia entre archivos	diff foo.c newfoo.c
file arch	Muestra el tipo de un archivo	file arc desconocido
find dir test acción	Encuentra archivos.	find . -name “_.bak” –print
grep [-cilnv] expr archivos	Busca patrones en archivos	grep mike /etc/passwd
at [-lr] hora [fecha]	Ejecuta un comando más tarde	at 6pm Friday < miscript
cal [[mes] año]	Muestra un calendario del mes/año	cal 1 2025
date [mmddhhmm] [+form]	Muestra la hora y la fecha	Date
kill [-señal] PID	Matar un proceso	kill 1234

Utilerías Unix

Editor de pantalla completa vi

El editor más frecuente en Unix es VI, Es un editor que trabaja línea a línea y que muestra una pantalla de texto a la vez.

Para iniciar una sesión de edición, se ejecuta el programa VI seguido del nombre del archivo a editar y dado el caso también la trayectoria. Si el archivo no existe VI lo crea. De igual manera, podemos simplemente invocar a VI, comenzar a escribir y después nombrar el archivo al momento de guardarlo.

Al ser ejecutado VI, presenta una pantalla con el texto del archivo y las líneas después del final del archivo aparecen con el carácter ~ para indicar que a partir de ahí el archivo está vacío. Si comenzamos a editar un archivo nuevo, todas las líneas aparecerán con éste carácter. Tiene tres modos de trabajo. El modo de inserción, el de edición y el de comando.

En el modo de inserción, toda la entrada que demos en el teclado se inserta en el archivo en el punto donde se encuentre el cursor. En el modo de edición daremos instrucciones que alteran el contenido, como por ejemplo para posicionarse en determinado punto, hacer reemplazos de texto, copiar o mover bloques de texto, etc. En el modo comandos se dan instrucciones para salvar el archivo, traer a edición otro, insertar otro archivo en el punto donde se está, terminar la edición, etc. En el modo edición, las instrucciones para mover el cursor en del texto son:

Comando	se desplaza:
l	un espacio a la derecha
h	un espacio a la izquierda
j	Una línea hacia abajo
k	Una línea hacia arriba
$	al final de la línea
^	al principio de la línea
w	a la siguiente palabra
e	al final de la palabra
b	al principio de la palabra
)	al final de la frase
(	al inicio de la frase
{	al inicio del párrafo
}	al final del párrafo
H	a la primera columna de la primera línea de la ventana
L	a la primera columna de la ultima línea de la ventana
nG	a la primera columna de la n-ésima línea del archivo

Para el control de la parte del texto que se despliega en la pantalla:

instrucción	Acción
^d	desliza el texto hacia arriba
^u	desliza el texto hacia abajo
^f	despliega la ventana de texto siguiente
^b	despliega la ventana de texto anterior
^l	redespliega el texto en la ventana actual

Las instrucciones para borrar texto:

Instrucción	Acción
dw	suprime la palabra donde está el cursor
dd	suprimir la línea donde está el cursor
D	borra el texto entre el cursor y el fin de la línea
x	borra el carácter sobre el que esta el cursor

Para hacer reemplazos de texto:

Instrucción	Acción
cw	cambiar la palabra actual
cc	cambiar la línea actual
C	cambiar desde el cursor hasta el final de la línea

r	cambiar el carácter sobre el que está el cursor

Algunas instrucciones suplementarias:

Instrucción	Acción
u	anular la última instrucción dada
/	realiza una búsqueda hacia delante
?	realiza una búsqueda hacia atrás
n	busca la siguiente ocurrencia de la última búsqueda
.	repite la última instrucción
Y	extrae la línea
p	se coloca en la línea de abajo
P	se coloca en la línea de arriba
ZZ	salva el archivo y termina la edición
ESC	cancela una orden
:	se cambia a modo comandos

Cuando vi esta en modo de edición, para cambiarse a modo inserción se hace con el carácter i y se posiciona antes del cursor y con a después de éste. Con Esc se cambia a modo edición de nuevo.

En modo de comandos, se tienen las siguientes funciones:

Instrucción	Acción
:w	salva el archivo en el disco
:q	abandona la edición sin guardar los cambios
:wq	escribe y termina
:q!	abandona sin escribir cuando se realizó algún cambio.
:r	carga otro archivo
:e	edita el archivo
:f	cambia o dá nombre al archivo actual
:n	se posiciona en la n-ésima línea
:Esc	se pasa a modo de edición

Existen además de vi otros editores para Unix, incluso más poderosos que él, pero esto depende de cada implementación de Unix. VI y ED son los únicos que se garantiza que se pueden encontrar en cualquier instalación de Unix. Entre los editores más populares se encuentra EMACS, pero cuenta con un conjunto de instrucciones realmente complejo, lo cual forma parte de su tradición de ser para usuarios avanzados.

Editor de línea ed

El editor ED fue hecho con la idea de tener un editor rápido y pequeño con lo mínimo indispensable. Es, además, un editor confiable y que puede ser usado en las peores condiciones: con terminales lentas, en conexiones por MODEM.

En su modo de operación normal, trabajan sobre una copia del archivo, para sobrescribirlo, hay que dar una instrucción específica. Trabaja sobre una línea o un grupo de líneas que cumplan con un patrón. Cada comando es un solo carácter, típicamente una letra. Cada comando puede ser precedido por uno o dos números de línea, que indican la línea o el rango de líneas al que serán aplicados. De no ser dado el número de línea, actúa sobre la actual. Veamos un ejemplo:

```
$ ed
a
Artifex vitae, artifex sui.

Muy cerca de mi ocaso yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
.
w en_paz
389
q
$
```

Después de invocar a ED inmediatamente le pedimos que añada el texto con el comando a y comenzamos a guardar un fragmento del texto de un pequeño poema. Terminamos poniendo un . como único carácter en la línea y después le pedimos que salve el archivo con el nombre de en_paz. ED nos responde con el número de caracteres que contiene y terminamos la sesión de edición con una q.

Si quisiéramos añadir unas cuantas líneas más, podríamos hacerlo con:

```
$ ed en_paz
389
a
. . . Cierto, a mis lozanías va a seguir el invierno;
mas tú no me dijiste que mayo fuese eterno!
.
q
?
w
489
q
$
```

En esta ocasión, invocamos a ED directamente con el nombre de un archivo que ya existe, así que nos responde con el número de carácteres que contiene el archivo. Comenzamos a añadir unas líneas, nos detenemos y tratamos de salirnos. Esta vez ed nos avisa con ? que el archivo no ha sido salvado. No debemos esperar una comunicación más comprensible de ED. Su manera de avisarnos que algo no le gusta o que las cosas no van bien es con un representativo ?. En este caso una segunda q le indicaría que realmente nos queremos salir sin salvar los cambios que hicimos. En todo momento una q le indica a ED que deseamos terminar la sesión sin guardar los cambios.

Cuando queremos revisar el texto ya escrito, le podemos pedir que imprima las líneas en un cierto rango con la instrucción a, bp donde a y b son números de línea. Por ejemplo, para ver el segundo párrafo:

```
$ ed angel
489
3,6p
Muy cerca de mi ocaso yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
.,$p
Porque veo al final de mi rudo camino
que yo fui el arquitecto de mi propio destino;
que si extraje las mieles o la hiel de las cosas,
fue porque en ellas puse hiel o mieles sabrosas;
cuando planté rosales coseché siempre rosas.
. . . Cierto, a mis lozanías va a seguir el invierno;
mas tú no me dijiste que mayo fuese eterno!
q
$
```

Los carácteres . y $ tienen el significado especial de ser la línea actual y la última línea respectivamente como podemos ver en la segunda parte del ejemplo. Si damos un enter por si sólo o un - seguido de un enter, nos lista la siguiente línea y la anterior respectivamente, moviéndose hacia adelante y hacia atrás en el archivo.

Podemos hacer búsquedas utilizando los operadores /patrón/ y ?patrón? hacia adelante y hacia atrás respectivamente. Una vez que encontramos el primero, podemos traer el siguiente con las formas // y ??.

```
$ ed angel
489
/fue/
fue porque en ellas puse hiel o mieles sabrosas;
//
mas tú
no me dijiste que mayo fuese eterno!
??
mas tú
no me dijiste que mayo fuese eterno!
q
$
```

Y podemos usar una búsqueda como parte de un rango:

```
1,/inmerecida/p
Artifex vitae, artifex sui.
Muy cerca de mi ocaso yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
.-1,.+3p
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
Porque veo al final de mi rudo camino
que yo fui el arquitecto de mi propio destino;
```

Y como vemos, también podemos usar el . como ancla e imprimir de la línea anterior a tres más adelante de la actual. La forma genérica de los comandos es número de línea o rango seguido del comando. Para añadir a partir de una línea se usa #a, para insertar antes de la línea #i. Para borrar el rango de líneas de la i-ésima a la j-ésima i, jd y i, jc para reemplazar el rango de líneas por las que se dan a continuación. Para hacer reemplazos en base a patrones se utiliza s/viejo/nuevo/:

```
3p
Muy cerca de mi ocaso yo te bendigo, Vida,
3s/ocaso/fin/
3p
Muy cerca de mi fin yo te bendigo, Vida,
```

Después del cambio se puede usar el modificador g para que haga el reemplazo en todas las ocurrencias en el rango de las líneas: s/viejo/nuevo/g.

Por supuesto que la sintaxis es extensible a rangos delimitados por patrones de búsqueda:

```
4,9s/viejo/nuevo/
1,$s/viejo/nuevo/
1,/viejo/s/viejo/nuevo/
```

Cuando necesitamos incluir el patrón viejo dentro del nuevo, no necesitamos reteclearlo, el carácter & toma el valor:

```
3p
Muy cerca de mi ocaso yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
3s/Vida/& mia/p
Muy cerca de mi ocaso yo te bendigo, Vida mia,
```

Y como vemos, la p después del comando de reemplazo imprime la línea. Para copiar y mover bloques, tenemos las instrucciones i, jtk i, jmk donde el primero copia las líneas en el rango de la i-ésima a la j-ésima después de la línea k-ésima. Igual en el segundo caso, pero borrándolas de su posición original.

También podemos incluir un archivo en el actual con la instrucción r en la forma nr archivo con la cual lo insertamos en el actual a partir de la línea n-ésima. Con i, jw archivo copiamos de la línea i-ésima a la j-ésima en el archivo denominado. Con i, jW archivo copiamos de la línea i-ésima a la j-ésima al final del archivo denominado.

Capítulo III. Bases de Datos

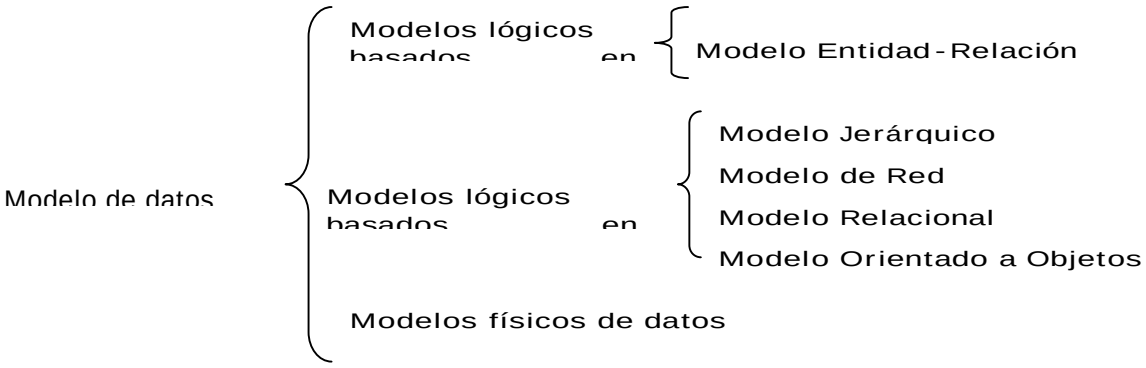
Modelos de Bases de Datos

Un modelo de Base de Datos es una colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender todas las necesidades de los diferentes usuarios.

Un modelo de datos consiste en:
- Objetos (entidades que existen y se manipulan)
- Atributos (características básicas de estos objetos)
- Relaciones (forma en que enlazan los distintos objetos entre si)

Un modelo de base 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. Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación.
- 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 cuatro modelos de datos más ampliamente aceptados son:
  - Modelo jerárquico.
  - Modelo de red.
  - Modelo relacional.
  - Modelo orientado a objetos.
- 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.



Evolución de los modelos de Bases de Datos

Época	Descripción
50's	Sistemas manejadores de archivos
60's	Surgen las bases de datos jerárquicas
70's	Surgen las bases de datos reticulares o de red
80's y 90's	Surgimiento y auge de las bases de datos relacionales
Últimos años	Surgimiento de las bases de datos orientadas a objetos

Descripción de los Modelos Lógicos Basados en registros

Modelo jerárquico

Una Base de datos jerárquica es un tipo de DBMS que almacenan la información en una estructura jerárquica que enlaza los registros en forma de estructura de árbol (similar a un árbol visto al revés), en donde un nodo padre de información puede tener varios nodos hijo.

Esta relación jerárquica no es estrictamente obligatoria, de manera que pueden establecerse relaciones entre nodos hermanos. En este caso la estructura en forma de árbol se convierte en una estructura en forma de grafo dirigido. Esta variante se denomina Bases de datos de red.

La forma de representar las relaciones y datos es 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 tenemos que, una base de datos jerárquica es una colección de árboles de este tipo.

Breve reseña histórica

Las bases de datos jerárquicas fueron concebidas en los años 1960. Uno de sus pioneros más importante es Charles Bachman. El primer metamodelo de base de datos propuesto fue la mencionada Base de datos en red, concebida bajo el auspicio de CODASYL (COnference on DAta SYstems Languages). Posteriormente se refinó la idea dando lugar a la base de datos jerárquica.

La primera implementación de este metamodelo fue IMS (Information Management System). Se trata de un diseño de IBM y otros colaboradores en 1966 para el Programa Apollo de la NASA. IMS aún se encuentra en activo.

El sector de la banca y las Administraciones Públicas adoptaron rápidamente esta tecnología. Estos sectores eran los únicos con capacidad económica suficiente para adquirir los enormes mainframe para la automatización de bases de datos, única solución posible en la época.

Poco después, en 1970, E. F. Codd propuso el modelo relacional. Las ventajas de este modelo y su enfoque matemático centraron los esfuerzos de la industria dando lugar a los sistemas gestores de bases de datos relacionales. Estos últimos han reemplazado a las bases de datos jerárquicas hoy día, pero no completamente. La mayoría de las antiguas bases de datos jerárquicas de bancos y Administraciones Públicas aún siguen en activo. Esto se debe a que el rendimiento de las bases de datos jerárquicas sigue sin ser superado por las bases de datos relacionales. Además estos sectores sufren un gran volumen de transacciones.

El modelo jerárquico no diferencia una vista lógica de una vista física de la base de datos. De manera que las relaciones entre datos se establecen siempre a nivel físico, es decir, mediante referencia a direcciones físicas del medio de almacenamiento (sectores y pistas).

Los datos se almacenan en la forma de registros, el equivalente a las filas del modelo relacional. Cada registro consta de un conjunto de campos, el equivalente a las columnas del modelo relacional. Un conjunto de registros con los mismos campos se denomina fichero (record type, en inglés), el equivalente a las tablas del modelo relacional.

El modelo jerárquico facilita relaciones padre-hijo, es decir, relaciones 1:N (de uno a varios) del modelo relacional. Pero a diferencia de éste último, las relaciones son unidireccionales. En justicia, dichas relaciones son hijo-padre, pero no padre-hijo. Por ejemplo, el registro de un empleado (nodo hijo) puede relacionarse con el registro de su departamento (nodo padre), pero no al contrario. La consulta en el sentido contrario requiere una búsqueda secuencial por todos los registros de la base de datos (por ejemplo, para consultar todos los empleados de un departamento). En las bases de datos jerárquicas no existen índices que faciliten esta tarea.

Obsérvese que, a priori, no existen relaciones N:M (de muchos a muchos) en el modelo jerárquico. Salvo que se simulen mediante varias relaciones 1:N. No obstante, esto puede provocar problemas de inconsistencia ya que el gestor de base de datos no controla estas relaciones.

Como ya se ha mencionado, las relaciones se establecen mediante punteros entre registros. Es decir, un registro hijo contiene la dirección física en el medio de almacenamiento de su registro padre. Esto tiene una ventaja fundamental sobre las bases de datos relacionales: el rendimiento. El acceso de un registro a otro es prácticamente inmediato sin necesidad de consultar tablas de correspondencia.

Las relaciones jerárquicas entre diferentes tipos de datos pueden hacer que sea muy sencillo responder a determinadas preguntas, pero muy difícil el contestar a otras.

Limitaciones del modelo jerárquico. A continuación se mencionan los problemas típicos de las bases de datos jerárquicas y que no existen en las bases de datos relacionales. Todos estos problemas derivan del hecho de que el sistema gestor de base de datos no implementa ningún control sobre los propios datos, sino que queda en manos de las aplicaciones garantizar que se cumplen las condiciones invariantes que se requieran (por ejemplo, evitar la duplicidad de registros). Dado que todas las aplicaciones están sujetas a errores y fallos, esto es imposible en la práctica. Además dichas condiciones suelen romperse ex profeso por motivos operativos (generalmente, ajustes debidos a cambios en el negocio) sin evaluarse sus consecuencias.

- Ø Duplicidad de registros. No se garantiza la inexistencia de registros duplicados. Esto también es cierto para los campos "clave". Es decir, no se garantiza que dos registros cualesquiera tengan diferentes valores en un subconjunto concreto de campos.
- Ø Integridad referencial. No existe garantía de que un registro hijo esté relacionado con un registro padre válido. Por ejemplo, es posible borrar un nodo padre sin eliminar antes los nodos hijo, de manera que éstos últimos están relacionados con un registro inválido o inexistente.
- Ø Desnormalización. Este no es tanto un problema del modelo jerárquico como del uso que se hace de él. Sin embargo, a diferencia del modelo relacional, las bases de datos jerárquicas no tienen controles que impidan la desnormalización de una base de datos. Por ejemplo, no existe el concepto de campos clave o campos únicos.

Modelo de red

Una base de datos de red como su nombre lo indica, esta formado por una colección de registros, los cuales están conectados entre sí por medio de enlaces. El registro es similar al de una entidad como las empleadas en el modelo entidad relación.

Un registro es una colección de campos (atributos), cada uno de los cuales contiene solamente almacenado un solo valor, el enlace es la asociación entre dos registros exclusivamente, así que podemos verla como una relación estrictamente binaria.

Una estructura de datos de red, llamada algunas veces estructura de plex, abarca más que la estructura de árbol porque un nodo hijo en la estructura red puede tener más de un padre. En otras palabras, las restricción de que un árbol jerárquico cada hijo puede tener un solo padre, se hace menos severa. Así, la estructura de árbol se puede considerar como un caso especial de la estructura 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.

Inconvenientes del Modelo en Red

Como hemos visto, el modelo en red tiene un carácter totalmente general. En el modelo en red no se ha especificado ningún tipo de restricción en lo que respecta a las interrelaciones. Esto quizá haga del modelo en red un modelo tremendamente sencillo de utilizar, pero no deja de tener un carácter general y provoca que en la práctica su instrumentación no resulte nada fácil. Es por esto, que los SGBD que se basan en el modelo en red, deben añadir una serie de restricciones a fin de poder implementar la base de datos físicamente y obtener un mayor rendimiento del sistema.

Un modelo de datos de este tipo, es el denominado modelo Codasyl. Este modelo es una simplificación del modelo en red general. Aquí solo se admiten ciertos tipos de interrelaciones y además se incluyen otras restricciones adicionales.

Modelo relacional

En este modelo se representan los datos y las relaciones entre estos, 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), cada uno de los cuales contiene solamente almacenado un solo valor, de cada registro localizado en la tupla. El enlace es la asociación entre dos registros exclusivamente, así que podemos verla como una relación estrictamente binaria En el modelo relacional el único elemento de representación es la tabla

Equivalencia entre representaciones

Representación Física	Representación Intuitiva	Modelo Relacional
Archivo secuencial	Tabla	Relación
Registros	Filas	Tuplas
Campos	Columnas	Atributos

- Ventajas
  - Facilita la comprensión y organización de un sistema de información, en términos de objetos.
  - Facilita documentar las reglas de negocio, si las organizamos y agrupamos en torno a los objetos con los que se relacionan.
- Desventajas:
  - No hay variedad de software que permitan el modelado y después su alimentación a diversos DBMS.
  - La desventaja anterior, obliga a usar herramientas no integradas; es decir, primero diseñar Objeto, por ejemplo y después elaborar manualmente el diseño derivado de los objetos en otra herramienta o a emular representación de Objetos semánticos bajo ER en diseñadores de tablas.

Modelo orientado a objetos

El modelo de bases de datos orientado a objetos es una adaptación a los sistemas de bases de datos. Se basa en el concepto de encapsulamiento de datos y código que opera sobre éstos en un objeto. Los objetos estructurados se agrupan en clases.

El propósito de los sistemas de bases de datos es la gestión de grandes cantidades de información. Las primeras bases de datos surgieron del desarrollo de los sistemas de gestión de archivos. Estos sistemas primero evolucionaron en bases de datos de red o en bases de datos jerárquicas y, más tarde, en bases de datos relacionales.

La interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes. Un método, es un trozo de código para implementar cada mensaje. Un método devuelve un valor como respuesta al mensaje.

En una base de datos orientada a objetos, la información se representa mediante objetos como los presentes en la programación orientada a objetos. Cuando se integra las características de una base de datos con las de un lenguaje de programación orientado a objetos, el resultado es un sistema gestor de base de datos orientada a objetos (ODBMS, object database management system). Un ODBMS hace que los objetos de la base de datos aparezcan como objetos de un lenguaje de programación en uno o más lenguajes de programación a los que dé soporte. Un ODBMS extiende los lenguajes con datos persistentes de forma transparente, control de concurrencia, recuperación de datos, consultas asociativas y otras capacidades.

Las bases de datos orientadas a objetos se diseñan para trabajar bien en conjunción con lenguajes de programación orientados a objetos como Java, C#, Visual Basic .NET y C++. Los ODBMS usan exactamente el mismo modelo que estos lenguajes de programación.

Los ODBMS son una buena elección para aquellos sistemas que necesitan un buen rendimiento en la manipulación de tipos de dato complejos. Los ODBMS proporcionan los costos de desarrollo más bajos y el mejor rendimiento cuando se usan objetos gracias a que almacenan objetos en disco y tienen una integración transparente con el programa escrito en un lenguaje de programación orientado a objetos, al almacenar exactamente el modelo de objeto usado a nivel aplicativo, lo que reduce los costos de desarrollo y mantenimiento.

El modelo de Base de datos relacional

Introducción

La siguiente tabla hace una síntesis de la evolución del Modelo Relacional, desde su surgimiento a fines de la década de los sesenta hasta la actualidad.

Años	Sucesos
1968	Surge el Modelo Relacional (Cood)
1970	Desarrollo Teóricos, algebra relacional (Codd, 1972)

1973	Prototipos (Ingres, Sistema R, etc.)
1979	Oracle
1981	SQL
1982	Sybase, Informix
1984	SQL / ANS
1986	SQL ISO
1990	Modelo Relacional Versión 2 (RM / V2) Codd
1992	SQL Estándar

El trabajo publicado por Codd en ACM (1970) presentaba un nuevo modelo de datos que perseguía una serie de objetivos, que se resumen en las siguientes líneas:

- Ø Independencia física. Es la capacidad de modificar el esquema físico sin provocar que se vuelvan a escribir los programas de aplicación. El esquema conceptual no es afectado por cambios al esquema físico de datos. Se refiere al ocultamiento de los detalles sobre las estructuras de almacenamiento a las aplicaciones de usuario o sea la descripción física de datos puede cambiar sin afectar a las aplicaciones de usuario. Si el DBMS modifica su organización interna de ficheros, no pasa nada con el esquema conceptual.
- Ø Independencia lógica. Capacidad de modificar el esquema conceptual sin provocar que se vuelvan a escribir los programas de aplicación. Algunos elementos del esquema externo no son afectados por cambios al esquema conceptual.
  - Ø Modificar columnas sólo afecta las vistas que incluyen esas columnas.
  - Ø Agregar más columnas no afecta las vistas.
  - Ø La creación de una nueva relación.
  - Ø El reordenamiento lógico de algunos atributos.
- Ø Flexibilidad. En el sentido de poder presentar a cada usuario los datos de la forma en que éste prefiera.
- Ø 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.
- Ø 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.

Entre las características más importantes de las bases de datos relacionales están:

- Ø En el enfoque relacional, los datos se organizan en tablas llamadas relaciones, cada una de las cuales se implanta como un archivo. En terminología relacional una fila en una relación representa un registro o una entidad. Cada columna en una relación representa un campo o un atributo.
- Ø Las entradas en la tabla tienen un solo valor (son atómicos); no se admiten valores múltiples, por lo tanto la intersección de un renglón con una columna tiene un solo valor, nunca un conjunto de valores.
- Ø Todas las entradas de cualquier columna son de un solo tipo. Cada columna posee un nombre único, el orden de las comunas no es de importancia para la tabla, las columnas de una tabla se conocen como atributos. Cada atributo tiene un dominio, que es una descripción física y lógica de valores permitidos.
- Ø Los registros poseen un campo identificador único (o combinación de campos) llamado llave primaria.
- Ø No existen 2 filas en la tabla que sean idénticas.
- Ø La información en las bases de datos son representados como datos explícitos, no existen apuntadores o ligas entre las tablas.
- Ø Así, todos los datos en una base de datos relacional se representan de una y sólo una manera, a saber, por su valor explícito (ésta se denomina en ocasiones “principio básico del modelo relacional”). En particular, las conexiones lógicas dentro de una relación y entre las relaciones se representan mediante esos valores.

Podemos decir que una Base de Datos Relacional es una Colección de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. 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.

Conceptos

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 deseamos guardar información. Estas se pueden clasificar en Fuertes o regulares y débiles:

- Ø Regulares: Son aquellas entidades que existen por sí mismas. La existencia de una entidad no depende de la existencia de otra entidad.
- Ø 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".

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.

Dominios. Se define dominio como un conjunto de valores que puede tomar un determinado atributo dentro de una entidad. De forma casi inherente al término dominio aparece el concepto restricción para un atributo. Cada atributo puede adoptar una serie de valores de un dominio restringiendo determinados valores.

Registro. Un registro es una colección de campos (atributos), cada uno de los cuales contiene solamente almacenado un solo valor.

Relaciones o Interrelaciones

El enlace es la asociación entre dos registros exclusivamente, así que podemos verla como una relación estrictamente binaria. Se entiende por interrelación a la asociación, vinculación o correspondencia entre entidades.

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.

En cada interrelación se debe establecer el número máximo y mínimo de ejemplares de un tipo de entidad que pueden estar asociadas, mediante una determinada relación con un ejemplar de la otra entidad. Este valor máximo y mínimo se conoce como cardinalidad y según corresponda, se representa de la siguiente forma: (0,N), (N,0), (1,N), (N,1), (0,1), (1,0) ó (N,N).

Redundancia

La redundancia de datos se refiere, a la existencia de información repetida o duplicada en diferentes tablas dentro de una base de datos. La redundancia conduce a muchos problemas que tienen que ver con la integridad y consistencia de los datos. La redundancia de los datos requiere múltiples procedimientos de entrada y actualización de los mismos.

Dentro de una base de datos relacional la redundancia debe ser mínima y controlada. 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 solo 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.

La inconsistencia entre dos entradas que representan al mismo "hecho" es un ejemplo de falta de integridad que, por supuesto, sólo ocurre si existe redundancia en los datos almacenados.

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

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

La integridad referencial. El término de integridad referencial se enmarca en la segunda regla de integridad y se aplica a las llaves foráneas: “Si en una relación hay alguna llave foránea, sus valores deben coincidir con valores de la llave primaria a la que hace referencia o bien, deben ser completamente nulos”. Es decir, las llaves foráneas no pueden dejar de tener correspondencia con la llave primaria de la tabla externa.

Llaves. Es un atributo o conjunto de atributos que permiten acceder a una sola tupla o a varias de la entidad. En el modelo entidad-relación al menos debe existir una llave que identifique de forma única a los registros (unicidad).

Llave 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 llave primaria para una determinada tupla será suficiente para identificarlo de manera única.

Llave candidata. Es el atributo o conjunto de atributos que podrían servir como llaves primarias. Una llave candidata debe cumplir dos condiciones:
- Ø Unicidad: no pueden existir dos tuplas con el mismo valor en todos los atributos que forman la llave candidata.
- Ø Minimidad: no existe ningún subconjunto de la llave que cumpla la regla de unicidad.

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

Llave foránea. Es una llave primaria en otra relación, éstas representan las asociaciones entre las diferentes entidades, es decir, son llaves que están siendo compartidas por dos tablas para formar una relación entre ellas. 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 en deterioro del tiempo de respuesta

Seguridad

Hoy en día se considera a la información de una empresa como uno de los activos más valiosos, 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.

Algunos sistemas operativos proporcionan algún nivel de seguridad en el control de acceso a usuarios, sin embargo ésta debe radicar principalmente en el SGBD o en la aplicación que maneje la base de datos, sobre todo para evitar la dependencia de entidades externas.

Es recomendable considerar las siguientes cuestiones:
- Ø Seguridad de Objetos. Se refiere a los permisos de los diferentes usuarios para poder hacer uso de tablas, procedimientos almacenados, triggers, etc.
- Ø Seguridad de operaciones. Aquí se manejan permisos para poder modificar (insertar, borrar, actualizar) la base de datos.

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. 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.

Normalización

El proceso de cristalización de las entidades y sus relaciones en formatos de tabla usando los conceptos relacionales se llama proceso de normalización y 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.

Las ventajas de la normalización son las siguientes:
- Ø Evita anomalías en inserciones, modificaciones y borrados.
- Ø Mejora la independencia de datos.
- Ø No establece restricciones artificiales en la estructura de los datos.
- Ø Están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

Las formas de normalización son:
- Ø 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.
- Ø 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 llave primaria es completamente dependiente de la llave primaria.
- Ø 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 llave primaria no depende transitivamente de la llave primaria. La dependencia es transitiva si existen las dependencias siendo atributos o conjuntos de atributos de una misma relación.

Reglas de Codd

En 1985 el Dr. Edgar Frank Codd publicó trece reglas para, si un sistema de bases de datos puede considerarse o no relacional.

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”. Cualquier cosa que no exista en una tabla no existe del todo. Toda la información, incluyendo nombres de tablas, nombres de vistas, nombres de columnas y los datos de las columnas deben estar almacenados en tablas dentro de las bases de datos. Las tablas que contienen tal información constituyen el Diccionario de Datos.

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”. Dado el valor de la clave primaria y dado el nombre de la columna requerida, deberá encontrarse uno y solamente un valor.

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”. La información de tablas, vistas, permisos de acceso de usuarios autorizados, etc., debe ser almacenada exactamente de la misma manera: En tablas. Estas tablas deben ser accesibles igual que todas las tablas, a través de sentencias de SQL.

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”. Esto significa

que debe haber por lo menos un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos.

6. Regla de actualización de vistas. “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 solo operador se aplica a la lectura, inserción, modificación y eliminación de datos”. Esto significa que las cláusulas SELECT, UPDATE, DELETE e INSERT deben estar disponibles y operables sobre los registros, independientemente del tipo de relaciones y restricciones que haya entre las tablas.

08. 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”.

09. 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.”
- Ø Ningún componente de una clave primaria puede tener valores en blanco o nulos. (esta es la norma básica de integridad).
- Ø Para cada valor de clave foránea deberá existir un valor de clave primaria concordante. La combinación de estas reglas aseguran que haya Integridad referencial.

11. Independencia de la distribución. “Un DBMS relacional tiene independencia de distribución”. El soporte para bases de datos distribuidas significa que una colección arbitraria de relaciones, bases de datos corriendo en una mezcla de distintas máquinas y distintos sistemas operativos y que esté conectada por una variedad de redes, pueda funcionar como si estuviera disponible como en una única base de datos en una sola máquina.

12. Regla de la no subversión. “Si un sistema relacional tiene un lenguaje de bajo nivel (un solo 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)”. Algunos productos solamente construyen una interfaz relacional para sus bases de datos no relacionales, lo que hace posible la subversión (violación) de las restricciones de integridad. Esto no debe ser permitido.

Sistema Administrador de Base de Datos Relacional

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, manejador de bases de datos (MBD) o en la mayoría de los casos, el sistema administrador de bases de datos DBMS (Data Base Management System). Un RDBMS es el conjunto de programas que permiten la definición, manipulación y control de acceso para una o varias bases de datos. Algunas características de los RDBMS son:
- Ø Facilitan la integridad, seguridad y acceso de los datos.
- Ø Los datos se almacenan como mínima redundancia.
- Ø Las aplicaciones son independientes del almacenamiento físico de los datos.

Un DBMS debe permitir las siguientes condiciones en una base de datos:
- Ø Los datos han de estar almacenados juntos.
- Ø Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento.
- Ø 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.
- Ø 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.
- Ø Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes DBMS.

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

- El usuario solicita alguna petición a la base de datos empleando algún sublenguaje de datos determinado (SQL).
- El RDBMS interpreta esa solicitud y la analiza.
- El RDBMS inspecciona en orden el esquema externo de ese usuario, la correspondencia externa/conceptual asociada, el esquema conceptual, la correspondencia conceptual/interna y la definición de la estructura de almacenamiento.
- El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada y devuelve una respuesta al usuario.

Arquitectura

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. Los componentes funcionales de un sistema de base de datos, son los siguientes:

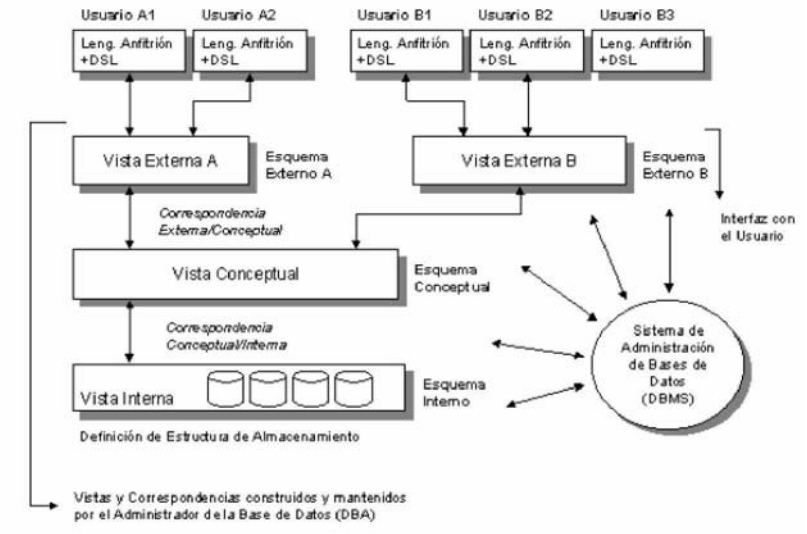
- Gestor de archivos.- Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar la información.
- Manejador de base de datos.- Sirve de interfaz entre los datos y los programas de aplicación.
- Procesador de consultas.- Traduce las proposiciones 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, estas 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:

- Nivel interno.- Es el nivel más bajo de abstracción y define cómo se almacenan los datos en el soporte físico, así como los métodos de acceso.
- Nivel conceptual.- Es el nivel medio de abstracción, se trata de la representación de los datos realizada por la organización, que recoge las vistas parciales de los requerimientos de los diferentes usuarios y las aplicaciones posibles, se configura como visión organizativa total e incluye la definición de datos y las relaciones entre ellos.
- Nivel externo.- Es el nivel de mayor abstracción, a este corresponden las diferentes vistas parciales que tienen de la base de datos los diferentes usuarios, en cierto modo, es la parte del modelo conceptual a la que tienen acceso.

La arquitectura relacional consta de los siguientes componentes:

- Modelo relacional de datos. En el nivel conceptual, el modelo relacional de datos está representado por una colección de relaciones almacenadas. Cada registro de tipo conceptual en un modelo relacional de datos se implanta como un archivo almacenado distinto.
- Sub-modelo de datos. Los esquemas externos de un sistema relacional se llaman sub-modelos relacionales de datos; cada uno consta de uno a más escenarios (vistas) para describir los datos requeridos por una aplicación dada. Un escenario puede incluir datos de una o más tablas de datos. Cada programa de aplicación está provisto de un buffer ("Área de trabajo de usuario") donde el DBMS puede depositar los datos recuperados de la base para su procesamiento o puede guardar temporalmente sus salidas antes de que el DBMS las escriba en la base de datos.
- Esquema de almacenamiento. En el nivel interno, cada tabla base se implanta como un archivo almacenado. Para las recuperaciones sobre las llaves principal o secundaria se pueden establecer uno o más índices para acceder a un archivo almacenado.
- Sub-lenguaje de datos. Es un lenguaje de manejo de datos para el sistema relacional, el álgebra relacional y cálculo relacional, ambos lenguajes son "relacionalmente completos", esto es, cualquier relación que pueda derivarse de una o más tablas de datos, también se puede derivar con un solo comando del sub-lenguaje. Por tanto, el modo de operación de entrada/salida en un sistema relacional se puede procesar en la forma de una tabla a la vez en lugar de un registro a la vez en otras palabras, se puede recuperar una tabla en vez de un solo registro con la ejecución de un comando del sub-lenguaje de datos.



Componentes de un RDBMS

DDL o Lenguaje de Definición de Datos: Se utiliza para crear, eliminar o modificar tablas, índices, vistas, triggers, procedimientos; es decir, nos permite definir la estructura de la base de datos mediante comandos como crear (Create), eliminar (Drop) o alterar (Alter).

- 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, triggers.

DML o 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. Los DML se distinguen por sus sublenguajes de recuperación subyacentes; se pueden distinguir dos tipos de DML, el procedural y el no procedural. La principal diferencia entre ambos es que en los lenguajes procedurales se tratan los registros individualmente, mientras que en uno no procedural se opera sobre un conjunto de registros. Las instrucciones relacionadas con este componente son:

- select. Permite realizar consultas a la base de datos.
- insert. Empleado para agregar registros a una tabla.
- update. Utilizado para modificar los valores de los campos de una tabla.
- delete. Utilizado para eliminar registros de una tabla.

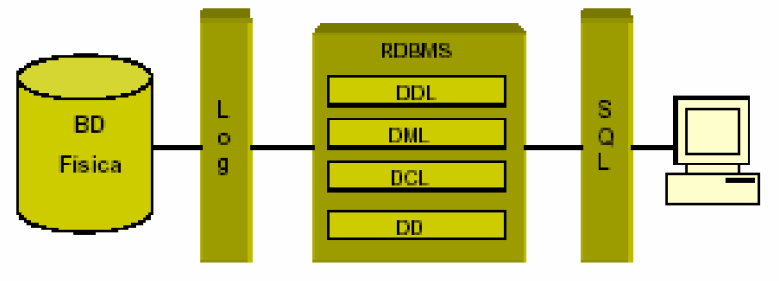
DCL o Lenguaje de Control de Datos: Se utiliza para la definición de los privilegios de control de acceso y edición a los elementos que componen la base de datos (seguridad), es decir, permitir o revocar el acceso. Los permisos a nivel base de datos pueden otorgarse a usuarios para ejecutar ciertos comandos dentro de la base o para que puedan manipular objetos y los datos que puedan contener estos. 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 o Diccionario de Datos: El contenido del diccionario puede considerarse como “datos acerca de los datos” (los cuales comúnmente reciben el nombre de metadatos), 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 pueda describir con facilidad que programas tienen probabilidad de ser afectados por un cambio propuesto al sistema. Sus principales funciones son las siguientes:

- Describe todos los elementos en el sistema.
- Los elementos se centran en los datos.
- Comunica los mismos significados para todos los elementos del sistema.
- Documenta las características del sistema.
- Facilita el análisis de los detalles para evaluar las características y determinar cómo deben realizarse los cambios.
- Localiza errores y omisiones del sistema.



Estándares Ansi

ANSI SQL 89

En 1989, tanto ANSI como ISO, publicaron estándares que definían al modelo relacional en el manejo de Bases de Datos. Estos estándares son: ANSI X3.135-1989 e ISO/IEC9075:1989.
Características principales:

- Agregan la capacidad conocida como integridad referencial y la descripción de todo el modelo relacional.
- Se definió que el lenguaje SQL está compuesto por comando, cláusulas, operadores. Estos elementos se combinan para definir y manipular la base de datos.
- Se establecen los elementos de un DBMS (DDL, DML y DCL), así como las instrucciones y sintaxis relacionadas con cada uno de ellos.
- Establecimiento de las cláusulas del comando select, las cuales son: From, Where, Group By, Having, Order By.
- Definición de los operadores lógicos: AND, OR y NOT.
- Definición de los operadores de comparación.
- Se determinan las funciones de agregado, tales como: AVG, COUNT, SUM, MAX, MIN.

ANSI SQL 92

Características principales:

- Toma todas características definidas en el estándar ANSI SQL 89.
- Permite la definición de esquemas.
- Permite la definición de dominios por parte de los usuarios, es decir, tipos de datos definidos por el usuario.
- Menciona las consideraciones para realizar consultas sencillas, multitablas y subconsultas.
- Incluye los operadores EXISTS y NOT EXISTS.
- Incorpora una nueva expresión de condición IS DISTINCT para la cláusula FROM.
- Menciona algunas consideraciones para el uso de las cláusulas GROUP BY y HAVING.
- Especifica la definición de vistas en una base de datos.

ANSI SQL 99

Características principales:

- Toma todas características definidas en los estándares ANSI SQL 89 y 92.
- Incluye nuevos tipos de datos escalares: BOOLEAN, CLOB (objeto de caracteres largo) y BLOB (objeto binario grande).
- Presenta dos nuevos operadores de totales: EVER y ANY.
- Incorpora generadores de tipo de dato: REF, ARRAY y ROW.
- Soporta una opción LIKE en CREATE TABLE, lo cual permite que la definicion de columna de una nueva tabla sean copiadas a partir de otra ya existente.
- Incluye la cláusula WITH para introducir nombres abreviados para determinadas expresiones.
- Incluye el objeto TRIGGER que ejecuta un STORE PROCEDURE como respuesta a un evento.

Marcas Comerciales de RDBMS

Microsoft SQL Server

Características principales:

- Ø Compatibilidad con estándares XML, Xpath, XSL, HTTP.
- Ø Obtiene código XML de las consultas realizadas con SQL.
- Ø Manipulación de documentos XML.
- Ø Manejo de bases de datos distribuidas.
- Ø Manejo de varias particiones físicas para almacenamientos de datos flexibles.
- Ø Permite realizar algunas tareas de mantenimiento y administración de la base de datos sin tener que darla de baja.
- Ø Permite realizar acciones OLAP (Online Analazing Processing), herramienta que permiten analizar datos en una Base da Datos, por medio de cubos de información.
- Ø Consulta y modificación de cubos virtuales de manera gráfica.
- Ø Conectividad con clientes ODBC y JDBC.

Sistema Operativo	Windows NT Server Windows 2000 Server (Diferentes Ediciones) Windows 2003 Server
Hardware	PIII o Superior, según los requerimientos del sistema
Memoria	Enterprise Edition: 64 MB como mínimo Standard Edition: 32 MB como mínimo
Disco Duro	Minima: 90 MB Máxima: 180
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 desea instalar

Sybase

Características principales:

- Ø Diseñado para soportar aplicaciones OLTP (On Line transaction Procesor, ambiente diseñado para insertar, actualizar y borrar datos en una base datos).
- Ø Permite integrar aplicaciones basadas en XML con la base de datos y crear reglas de negocio que ejecuten Java Beans.
- Ø Conectividad con clientes ODBC y JDBC.
- Ø Soporte para BLOB's (Large Objects).
- Ø 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.
- Ø Tamaño expandido de filas y datos, es decir soporta filas más 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)
- Ø Compresión de copias de respaldo.
- Ø Bloqueo a: nivel de fila, nivel de pagina de datos, nivel de página (datos e índices), nivel de tabla.

Fabricante
Sun Microsystem
Hewlet Packard
IBM
SGI
Compaq

Sistema Operativo	Memoria
AXP	94 MB
HP: 32 / 64	64 MB / 90 MB
LINUX	32 MB
Windows (NT, 2000, 2003) 32 y 64	46 MB
Sun: 32 / 64	66 MB / 93 MB
Especificaciones del Servidor	
Capacidad Disco Duro	240 MB
Tamaño de la Base de Datos	32,767
Bases de Datos en Update	4 TB
Tablas en una consulta	16
Usuarios por Base de Datos	50
Columnas por Tabla	2,146,484,223
Tamaño de la Pagina	1024
Argumentos por Procedimientos	2K, 4K, 8K, 16K, 1024

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.

Sistema Operativo	Memoria
Solaris	94 MB
HP - UX	64 MB
Compaq Tru64	90 MB
AIX	32 MB
HP Alpha	46 MB
Linux	66 MB
Windows (NT, 2000, 2003) 32 y 64	93 MB

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 quieries 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.

Sistema Operativo	Memoria
IBM AIX	94 MB
SGI IRIX	64 MB
Sun Solaris	90 MB
HP UX	32 MB
Compaq Tru64	46 MB
Linux	66 MB
Windows (NT, 2000, 2003) 32 y 64	93 MB

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.

Sistema Operativo
Cualquier equipo y sistema operativo UNIX / LINUX

MySQL
Características principales:
Ø Soporta los estándares ANSI.
Ø Contiene esquemas de almacenamiento independiente que se pueden seleccionar de acuerdo a las necesidades.
Ø InnoDB para transacciones y bloqueo de registros.
Ø MyISAM sin transacciones
Ø Soporte para SSL.
Ø Querys con manejo de cache que puede incrementar el performance de la base de datos en un 200%.
Ø Permite manejo de replicación de bases de datos.
Ø Soporta indexado de texto.

Sistema Operativo
Linux, Windows, FreBSD, Sun Solaris, IBM – AIX, MAC OS X, HP – UX

Clasificacion de RDBMS por Tipos de Licencia:
Ø Por usuario conectado.
Ø Por procesador.
Ø Software Libre (PostgresSQL, Mysql, sybase (Linux)).

Por volumen de información:
Ø Corporativo (Oracle, Informix, Sybase, DB2, PostgreSQL)
Ø Departamental (SQL Server, SQL Anywhare, Mysql).

Programación de la Base de Datos. Transact-SQL

Introducción

SQL (Structured Query Language; Lenguaje Estructurado de Consulta) es un lenguaje de consulta para bases de datos, 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 DataBase Management System) 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, Interbase, 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.

La información presentada en este tema con respecto a los manejadores de bases de datos relacionales está basada en las versiones: MySQL 4.1, PostgreSQL 8.0, Oracle 10i, Sybase XI, MS SQL Server 2000

Definición de datos.

Antes de comenzar a trabajar con SQL, es necesario conocer los elementos que intervienen en la definición de la información en una base de datos, para poder manipularla de manera adecuada.

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.

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. Por ejemplo MS SQL Server acepta el tipo de dato DOUBLE PRECISION pero lo convierte y maneja como un FLOAT

En la siguiente tabla se muestra en la primera columna, el tipo de dato como se especifica en el ANSI SQL y en las demás columnas se indica el tipo de datos equivalente que cumple con dicho estándar.

Tipo en SQL99	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server	Descripción
tinyint	tinyint			tinyint	tinyint	Entero de 1 byte
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
Blob	Blob	bytea	blob	image	image	Binary large object
Clob	Text	text	clob	text	text	Character large object
boolean		boolean		bit	bit	Valor booleano
Interval		interval				Intervalo de tiempo

En algunos casos existe en el manejador el nombre del tipo de dato estándar, más no cumple con las especificaciones que debería

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.

Las operaciones de agregado (AVG, SUM, etc.) se da cuenta de los valores nulos y no los considera en el calculo. Sin embargo, en las operaciones matemáticas o de texto si se da cuenta y regresa un valor NULO.

Tablas

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.

Los nombres que se le asignan a los objetos de una base de datos, están sujetos a ciertas reglas que varían entre RDBMS, incluso de una versión a otra del mismo. A continuación se listan algunas de las reglas que deben observarse:

- Ø No pueden usarse palabras reservadas del servidor SQL.
- Ø Es aconsejable usar nombres descriptivos.
- Ø No pueden existir dos objetos (aunque sean de distinto tipo) con el mismo nombre para un usuario en particular
- Ø Deben empezar con una letra.
- Ø Pueden contener letras: A-Z, a-z, números: 0-9, _, $

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

```
CREATE TABLE <nombre_tabla>(
<nom_campo1> <tipo_dato> [NULL | NOT NULL] [DEFAULT <val_predetermi>],
. . .
<nom_campoN> <tipo_dato> [NULL | NOT NULL] [DEFAULT <val_predetermi>])
```

Dependiendo del RDBMS la modificación de estructura de la tabla ofrece mayor o menor flexibilidad. Por ejemplo, MySQL es de las más flexibles, porque permite cambiar incluso el tipo de dato de una columna mientras que en Sybase o MS SQL Server es necesario eliminar y volver a crear la tabla con la estructura deseada.

Características soportadas:	MySQL	PostgreSQL	Oracle	Sybase	MS SQL Server
Agregar campo	Si	Si	Si	Si	Si
Eliminar campo	Si	Si	Si	No	Si
Renombrar campo	Si	Si	Si	No	No
Cambiar el tipo de dato del campo	Si	No	Si	No	Si
Definir llave primaria	Si	Si	Si	Si	Si
Definir llave foránea	Si	Si	Si	Si	Si
Renombrar tabla	Si	Si	Si	No (pero brinda un procedimiento almacenado sp_rename)	No (pero brinda un procedimiento almacenado sp_rename)

La sintaxis general para agregar un campo es la siguiente:

```
ALTER TABLE <nombre_tabla> add <nombre_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

En la práctica, muchas de estas reglas no se definen en la base de datos sino mediante la lógica de una aplicación desarrollada en algún lenguaje, que tiene acceso a la información ya

que el exceso de reglas puede disminuir el rendimiento del RDBMS en los procesos de inserción y modificación de información.

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.

	MySQL	PostgreSQL	Oracle	Sybase	MS SQL Server
Soporta definición de reglas	No	Si	Si	Si	Si

Sintaxis para Sybase y Ms SQL Server:
```
CREATE RULE <nombre_regla>
AS <condicion>
sp_bindrule regla_salario, 'empleado.sueldo'
```

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.

	MySQL	PostgreSQL	Oracle	Sybase	MS SQL Server
Soporta definición de defaults	Si	Si	Si	Si	Si

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.columna>
```

Llaves e Índices

Un índice es una estructura de almacenamiento físico que permiten recuperar datos de una manera muy eficiente.

Es un error frecuente confundir entre índices y llaves, quizá sea el hecho de que al crear una llave primaria se genera un índice, lo cual no sucede para una llave foránea.

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.

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 varias formas de declarar una llave primaria. Al crear una tabla se puede especificar que campo o campos forman parte de la llave primaria.
```
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) )
```

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)
```

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. En el momento de crear la tabla:

```
CREATE TABLE partido (
id_pais_local NUMBER(3) NOT NULL, id_pais_visitant NUMBER(3) NOT NULL,
 marcador   VARCHAR2(15) NOT NULL,
 CONSTRAINT pais_01_fk FOREIGN KEY (id_pais_local) REFERENCES
pais(id_pais),
 CONSTRAINT pais_02_fk FOREIGN KEY (id_pais_visitante) REFERENCES
pais(id_pais));
```

A través de un ALTER TABLE:

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

Para crear índices se utiliza el siguiente comando:

```
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. Por lo general se suele crear un índice únicamente cuando una columna es empleada frecuentemente en una cláusula WHERE y la tabla tiene un tamaño considerable.

Para eliminar un índice se utiliza la siguiente instrucción:

```
DROP INDEX <nombre_índice>
```

Manipulación de datos.

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

- Ø Select. Permite seleccionar (recuperar) información de una tabla.
- Ø Insert. Permite agregar información a una tabla.
- Ø Delete. Permite eliminar información de una tabla.
- Ø Update. Permite actualizar información que existe en una tabla.

Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples. En caso contrario será interpretado como 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.

```
SELECT {* | [DISTINCT] <campo>, <campo> ... }
FROM <nombre_tabla>, [ <nombre_tabla> ]
[WHERE <condición> ]
[GROUP BY <campo>, <campo>,... ]
[HAVING <condición> ]
[ORDER BY <campo> [ASC|DESC], <campo> [ASC|DESC],... ]
```

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.

La operación de selección genera un subconjunto de los renglones de una tabla, con base en un criterio (restricción) establecido. Esta es realizada por la cláusula WHERE de SQL.

La proyección selecciona y genera un subconjunto con los atributos (columnas) indicados de una tabla. También es conocida como operación vertical. Esta operación es realizada por la cláusula SELECT del SQL.

La operación unión realiza la misma acción que en el álgebra de conjuntos. Esta operación se realiza con la cláusula UNION del SQL.

El producto cartesiano es el producto cruz entre 2 tablas. El resultado es la combinación de cada renglón de una tabla con cada renglón de la otra tabla. En SQL esta operación se lleva a cabo cuando se ocupa la cláusula FROM especificando 2 o más tablas y no se especifica una restricción que indique la relación entre las tablas.

La operación Join es en esencia un producto cartesiano, donde se selecciona los renglones que satisfagan las condiciones indicadas que establecen la relación entre las tablas involucradas. Esta es una operación muy común en las bases de datos relacionales.

SELECT nos permite indicar el nombre de los campos que queremos 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>. O se puede colocar inmediatamente después del nombre de la tabla: <tabla> [seudónimo]

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. Igualmente, si el nombre del campo es ocupado por varias tablas en la consulta es necesario identificar a que tabla pertenece.
<nombre tabla>.<nombre_campo>

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.

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 is NULL;
```

Las expresiones más frecuentes son las que involucran una comparación entre dos elementos.

Expresión	Sintaxis
Igualdad	=
Desigualdad	<> ó !=
Mayor que	>
Menor que	<
Mayor o igual que	>=
Menor o igual que	<=
Similar a	LIKE
Es	IS
No es	IS NOT
En una Lista A, B, C, ...	IN (A, B, C, ...)
No en una Lista	NOT IN (...)
Entre A y B	BETWEEN A AND B

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

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:

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

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.

ORDER BY 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.

```
ORDER BY <Campo> [ DESC | ASC ]
ORDER BY <Posicion del Select> [ DESC | ASC ]
```

Joins Internos.

Un join se emplea para definir la relación que existe entre dos tablas. El INNER JOIN es una cláusula que nos permite definir estas relaciones. Este tipo de join puede expresarse de otra manera utilizando restricciones en la sección del WHERE en una consulta:

```
SELECT e.nombre, d.nombre FROM empleado e INNER JOIN departamento d ON
e.id_departamento = d.id_departamento
```

La consulta anterior puede escribirse sin utilizar el INNER JOIN de la siguiente manera:

```
SELECT e.nombre, d.nombre FROM empleado e, departamento d WHERE
e.id_departamento = d.id_departamento
```

El resultado de las dos consultas mostradas es el mismo. Lo único que varía es la forma de especificar la relación entre las tablas.

Joins Externos

Si en la relación de tablas una de ellas no tiene valor en la columna de liga estos registros no se presentarán en el SELECT. Es en estos casos donde es de gran utilidad el uso de joins externos, que permiten ignorar esta falta de correspondencia. El join externo se maneja de manera distintas en cada RDBMS. Por lo tanto para mostrar el listado de manera adecuada, se emplearían las siguientes consultas:

En Sybase ó Ms SQL Server:

```
SELECT e.nombre, d.nombre FROM empleado e, departamento d WHERE
e.id_departamento *= d.id_departamento
```

En Oracle:

```
SELECT e.nombre, d.nombre FROM empleado e, departamento d WHERE
e.id_departamento = d.id_departamento(+)
```

En MySQLó PostgreSQL:

```
SELECT e.nombre, d.nombre FROM empleado e LEFT JOIN departamento d ON
e.id_departamento = d.id_departamento
```

Oracle adicionalmente brinda las operaciones de intersección (INTERSECT) y la de resta (MINUS).

Union. Permite realizar la unión del resultado de dos consultas. El proceso de UNION elimina los registros duplicados, aunque algunos RDBMS proporcionan una cláusula que permite mostrar todos los registros resultantes de la unión. Ejemplo: Mostar el sueldo máximo y el mínimo que perciben los empleados.

```
SELECT MAX(sueldo) FROM empleado
UNION
SELECT MIN(sueldo)FROM empleado
```

Inserción de datos.

A través de la instrucción INSERT de SQL, se introduce la información a una tabla. Esta cláusula permite indicar que la operación a realizar es la inserción de un registro. La estructura general de este comando es la siguiente:

```
INSERT INTO <tabla> [(<nombrCampo1>, <nombrCampo2>, <nombrCampo3> .)]
{VALUES (<valorCampo1>, <valorCampo2>, <valorCampo3> ... ) | <Expresión
select> }
```

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, por que 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. La Cláusula VALUES permite especificar los valores a insertar para cada uno de los campos involucrados en la sentencia

Eliminación de datos.

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> ]
```

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> ]
```

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.

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.

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.

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.

Aunque las vistas forman parte del estándar, algunos RDBMS como MySQL no las implementan actualmente. Las vistas pueden ser utilizadas como si fueran tablas, pero con ciertas restricciones. La sintaxis para crear una vista es:

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

Para eliminar una vista se utiliza la siguiente instrucción.
```
DROP VIEW <nombre_vista>
```

La instrucción SELECT que define una vista no puede utilizar la cláusula ORDER BY. Sin embargo, la cláusula ORDER BY se puede aplicar a la vista una vez creada.

Programación

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. 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-END IF

La sintaxis en Sybase, MS SQL Server, Oracle es:
```
IF <expresión>
<bloque A de instrucciones>
ELSE <bloque B de instrucciones>
```

Para Oracle se termina con un END IF. Si la expresión evaluada es verdadera se ejecuta el bloque A de instrucciones y de lo contrario se ejecuta el bloque B.

IF-THEN-ELSE-END IF

La sintaxis en Sybase, MS SQL Server es:
```
IF <expresión 1>
<bloque A de instrucciones>
ELSE IF <expresión 2>
<bloque B de instrucciones>
ELSE IF <expresión 3>
<bloque C de instrucciones>
ELSE
<bloque D de instrucciones>
```

Para Oracle se termina con un:
END IF

La sintaxis en Oracle únicamente difiere en la cláusula ELSE IF, la cual se escribe como ELSIF.

FOR LOOP

Este ciclo iterativo inicializa la variable especificada y ejecuta las instrucciones cíclicamente hasta que la variable alcanza el valor final. La sintaxis en Oracle es la siguiente:
```
FOR <variable> IN <valor inicial>..<valor final> LOOP
 <lista_de_instrucciones>
END LOOP;
```

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;
```

Etiquetas

Aunque las etiquetas no se recomiendan emplearlas en un ambiente de programación estructurado, pueden ser muy útiles para resolver un problema de una manera muy sencilla. Las etiquetas en Sybase se emplean junto con la instrucción GOTO para trasladar el flujo del programa a la línea donde se encuentra la etiqueta señalada por el Goto. La sintaxis es:

```
  GOTO <etiqueta>
.
```

```
  <etiqueta>:
```

Por ejemplo :

```
  DECLARE @contador int
  SELECT @contador = 0;
  inicia:
     SELECT @contador = @contador + 1;
     PRINT "Ciclo numero " + convert(varchar, @contador);
     IF @contador <= 10
     BEGIN
        GOTO inicia
     END
```

Procedimientos almacenados.

Un procedimiento almacenado es un conjunto de comandos de SQL que son 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. El lenguaje que se emplea para programar los procedimientos almacenados varía de un RDBMS a otro y existen algunos que permiten programar en más de un lenguaje.

Declaración de variables.

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:

- Ø Los nombres de las variables locales deben estar precedidos por una @
- Ø 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)> [, .....]
```

Para asignar valores a una variable se utiliza la instrucción SELECT, de la siguiente forma:

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

En este caso el uso de las cláusulas FROM y WHERE son opcionales. Por ejemplo:

```
  -- asignando constantes a las variables
  SELECT @descuento = 0.45, @fecha = getdate()
  -- asignando valores de una tabla a las variables
  SELECT @salario = sueldo, @nombre = nombre FROM empleados WHERE
id_empleado = 10
```

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

Sybase y Ms SQL Server utiliza una serie de variables para alojar información a distintos niveles ya sea a nivel de servidor, a nivel de sesión o a nivel de proceso; a dichas variables se les llama variables globales y el usuario no puede declararlas o asignarles valores. De esto se encarga automáticamente el servidor. Las variables globales van precedidas por doble @.

Algunas de las variables globales más útiles son:

Variable	Contiene
@@error	número de error generado por el último comando
@@rowcount	número de registros afectados en el último comando
@@version	número de versión
@@spid	número de proceso en el SQL server
@@servername	nombre del servidor
@@sqlstatus/@@fetch_status (Sybase/MS SQL Server)	estatus del último comando fetch en un cursor 0/0 Éxito 1/-1 Error 2/-2 No más registros

En algunos casos, es posible que se desee que el tipo de una variable coincida con el tipo usado para una columna de una tabla determinada, en esos casos se puede usar la construcción:

```
DECLARE sueldo empleado.sueldo %TYPE;
```

Con lo cual se logra que la variable sueldo tenga el mismo tipo que la columna sueldo de la tabla empleado. También es posible inicializar las variables, mediante el operador: =. Además, mediante el uso del mismo operador es posible hacer asignaciones en el cuerpo del programa. Por ejemplo:

```
DECLARE salario NUMBER:= 15000;
BEGIN
 salario := salario + 1500;
END;
```

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
```

En Oracle no difiere mucho la sintaxis:

```
CREATE PROCEDURE <nombre_procedimiento>
[(<nombre_parámetro> [IN | OUT] <tipo_de_dato>) ]
AS
<instrucciones SQL>
```

Ejecución de procedimientos. Para ejecutar un procedimiento almacenado en Sybase o Ms 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>, ...]
```

En el caso de Oracle se emplea la instrucción CALL para llamar al procedimiento almacenado:

```
CALL nombre_empleados
```

Eliminación de procedimientos.

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

```
DROP PROCEDURE <nombre_procedimiento>
```

Triggers

Características.

Un trigger es un procedimiento almacenado que es invocado cuando un evento en particular ocurre. Los procedimientos almacenados que se invocan tiene la restricción de que no pueden manejar parámetros ni ser invocados directamente.

En Sybase y MS SQL Server un trigger al dispararse crea dos tablas temporales 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

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.

Para crear un trigger se utiliza la siguiente sintaxis:

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

... RETURN	

Con las cláusulas BEFORE y AFTER se especifica en que momento debe de activarse la acción, si antes de la realización del evento o después. A un lado de cualquiera de estas dos cláusulas, se especifica el evento que va disparar la acción, pudiendo ser INSERT, DELETE o UPDATE.

En el caso de una actualización es posible especificar que únicamente cuando la modificación afecte a cierta columna se dispare la acción. Esto se realiza empleando la cláusula OF COLUMN seguida del nombre de la columna.

La cláusula ON permite especificar en que tabla se va a asociar el disparador de acción. Después de la cláusula ON se coloca el nombre de la tabla.

La cláusula FOR EACH_ROW permite especificar que la acción se ejecutara a nivel renglón. Es decir por cada registro afectado por la inserción, modificación o eliminación se realizara la acción especificada en el trigger.

Eliminación. La sintaxis para eliminar un trigger es la siguiente:
DROP TRIGGER <nombre_trigger>

Cursores.

Los cursores son apuntadores que tienen acceso de manera individual a un registro dentro de un conjunto de resultados, provenientes de una consulta a la base de datos. Los cursores se emplean dentro de procedimientos almacenados cuando se necesita realizar operaciones complejas de selección o actualización de datos o cuando se necesita realizar operaciones que afecte registro por registro.

Uso de Cursores. Al utilizar cursores se deben de seguir los siguientes pasos:
- Ø 1.- Declaración del cursor
- Ø 2.- Apertura del cursor
- Ø 3.- Recorrer los registros del cursor (realizar procesos)
- Ø 4.- Cerrar el cursor

Declaración del cursor. En la declaración de un cursor se indica: el nombre del cursor y una instrucción SELECT la cual definirá los campos y registros que se consultarán a la base de datos. Para declarar un cursor se sigue la siguiente sintaxis:

Sybase/MS SQL Server	Oracle
DECLARE <nombre_cursor> CURSOR FOR <instrucción Select> [FOR {READ ONLY | UPDATE [OF <campos>] }]	DECLARE CURSOR <nombre_cursor> IS <instrucción Select>;

Apertura del cursor

Un cursor internamente es una sentencia SELECT cuyo resultado se guarda en el servidor en tablas temporales y que se va retornando cada una de las filas según se va pidiendo desde el cliente.

El primer paso es ejecutar la sentencia SELECT y guardar su resultado dentro de las tablas temporales. Este paso se denomina Abrir el cursor. La apertura del cursor debe realizarse sólo una vez. La sintaxis de apertura de un cursor es:

Sybase/MS SQL Server	Oracle
OPEN <nombre_cursor>	OPEN <nombre_cursor>;

Una vez que el cursor está abierto, se podrá empezar a pedir los resultados al servidor.

Recorrer los registros.

Una vez que el cursor está abierto en el servidor se podrá hacer la petición de recuperación de fila. Este paso es equivalente a hacer una consulta SELECT de una sola fila ya que estamos seguros de que no vamos a recuperar más de una fila. La sintaxis de recuperación de fila de un cursor es:

Sybase/MS SQL Server	Oracle
FETCH <nombre_cursor> INTO <variables>	FETCH <nombre_cursor> INTO <variables>;

Podremos recuperar filas mientras la consulta SELECT tenga filas pendientes de recuperar. Para saber cuándo se debe detener el ciclo de instrucciones FETCH, se realiza lo siguiente:

Sybase/MS SQL Server	Oracle
Se evalúa la variable @@sqlstatus/@@fetch_status respectivamente que contiene los siguientes valores: 0/0 Se logró el acceso al registro 1/-1 No se logró el acceso al registro 2/-2 No hay más registros	Se realiza la evaluación de alguno de los siguientes atributos: <nombre_cursor>%FOUND BOOLEAN. Retorna si la última fila recuperada fue válida <nombre_cursor>%ISOPEN BOOLEAN. Retorna si el cursor está abierto <nombre_cursor>%NOTFOUND BOOLEAN. Retorna si la última fila fue inválida <nombre_cursor>%ROWCOUNT NUMBER. Retorna el número de filas recuperadas

Así lo acción más típica es recuperar filas mientras queden alguna por recuperar en el servidor. Esto se logra través del siguiente bloque de código:

Sybase/MS SQL Server	Oracle

FETCH <nombre_cursor> INTO <@variables>
WHILE (@@sqlstatus = 0)
BEGIN
<instrucciones para procesar cada registro>
/* Avanza al siguiente registro */.
FETCH <nombre_cursor> INTO <@variables>
END
	LOOP
	FETCH <nombre cursor> INTO <variables>;
	EXIT WHEN <nombre_cursor>%NOTFOUND;
	<instrucciones para procesar cada registro>
	END LOOP;

Cierre del cursor. Finalmente para liberar los recursos empleados por el cursor, este debe ser cerrado. La sintaxis para liberar el cursor es:

Sybase/MS SQL Server	Oracle
DEALLOCATE [CURSOR] <nombre_cursor>	CLOSE <nombre_cursor>;

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.

En Sybase:
Antes de que un usuario pueda acceder a alguna base de datos, éste debe primero darse de alta como usuario en el servidor. Para esto se usa el siguiente procedimiento almacenado, teniendo activa la base de datos MASTER:
sp_addlogin <nombre_usuario> , [<contraseña>] , [<base_de_datos_default>]

Posteriormente debe darse de alta al usuario en la base de datos que se desea pueda acceder; para esto se activa la base de datos deseada y se utiliza el siguiente procedimiento almacenado para darlo de alta:
sp_adduser <nombre_usuario>

Se pueden controlar los privilegios a los usuarios de creación de objetos en la base de datos, así como la manipulación de la información contenida en ella, mediante la siguiente sintaxis:
GRANT <privilegio> TO <usuario>
REVOKE <privilegio> FROM <usuario>

Donde GRANT se utiliza para otorgar privilegios y REVOKE para eliminarlos.

En PostgreSQL: El esquema del manejo de usuarios y permisos no es tan refinado como en otros RDBMS. Los usuarios se crean con la siguiente sintaxis:
CREATE USER <nombre_usuario>[WITH { PASSWORD <contraseña> | CREATEDB | NOCREATEDB | CREATEUSER| NOCREATEUSER }]

En MySQL: No existe una sintaxis propia para crear usuarios sin embargo en el momento en el que se otorgan permisos, si el usuario al que se refiere la instrucción no existe, entonces es creado.
GRANT ALL ON BaseDeDatos. TO NuevoUsuario IDENTIFIED BY 'Contraseña';*

En Oracle:
La sintaxis básica para crear un nuevo usuario es la siguiente:
CREATE USER <nombre_usuario> IDENTIFIED BY <contraseña>;

Una vez creado el usuario es necesario establecer los permisos correspondientes con la cláusula GRANT, para que se pueda utilizar dicha cuenta.

Funciones de utilidad.

Aunque el estándar SQL92 define las funciones con las que debe contar un RDBMS, desafortunadamente en la práctica, no todos los RDBMS cumplen con el estándar.

A continuación se muestran aquellas funciones de uso frecuente. Los tipos de carácter soportados para SQL92 son char, varchar y text.

Función de Tipo Carácter	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

Función Comunes SQL92	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server
char_length	Si	Si	No	No	No
character_length					
lower	Si	Si	Si	Si	Si
octet_length	Si	Si	No	No	No
position	Si	Si	No	No	No
substring	Si	Si	No estándar	No estándar	No estándar
trim	Si	Si	Si	No	No
upper	Si	Si	Si	Si	Si
||	No	Si	Si	No	No
Case	Si	Si	Si	No	No

A continuación se muestran las funciones de Sybase/Ms SQL Server que remplazan a las ausentes del estándar SQL92:

FUNCION	RDBMS	SINTAXIS
substring	SYBASE/ MSSQL	substring (<cadena>,#empiezo, #longitud) Extrae una cadena de la cadena a partir de #empiezo hacia la izq. de #longitud
charindex	SYBASE/ MSSQL	charindex (<cadena1>,<cadena2>) Regresa la posición donde se encuentra la cadena1 dentro de la cadena2

Ltrim	SYBASE/ MSSQL	ltrim (<cadena>) Elimina los blancos a la izquierda de la cadena
Rtrim	SYBASE/ MSSQL	rtrim (<cadena>) Elimina los blancos a la derecha de la cadena
datalength	SYBASE/ MSSQL	datalength (<cadena>) Regresa la longitud de la cadena
Concat	MySQL	concat(<cadena1>, <cadena2>) Realiza la concatenación de las cadenas de texto especificadas
Length	Oracle	Length(<cadena>) Obtiene la longitud de la cadena de texto especificada.
Instr	Oracle	Instr(<cadena1>,<cadena2>) Busca la cadena2 dentro de la cadena1, regresando la posición donde se encontró.
Substr	Oracle	Substr(<cadena>,#empiezo, #longitud) Extrae una cadena de la cadena a partir de #empiezo hacia la izq. de #longitud (Si este último se omite, se regresa hasta el final de la cadena)

A continuación se muestra un listado de las funciones matemáticas que forman parte del estándar SQL92 y en su caso, la función equivalente.

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

Las funciones estándar para manejo de fecha son las siguientes:

Función	Descripción	MySQL	Pstrgre SQL	Oracle	Sybase	MS SQL Server
Current date / Current_date()	Regresa la fecha actual en formato 'YYYY-MM-DD' o YYYYMMDD	Si	Si	Sysdate	Getdate()	Getdate()
Current time / Current_time()	Regresa la hora actual en formato 'HH:MM:SS' o HHMMSS	Si	Si	Sysdate	Getdate()	Getdate()
Current_timestamp / Current_timestamp()	Regresa la fecha y hora actual con formato 'YYYY-MM-DD HH:MM:SS' o YYYYMMDDHHMMSS	Si	Si	Sysdate	Getdate()	GetDate()

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.

Capítulo IV. Temas Avanzados de Bases de Datos

Carga y Configuración de Bases de Datos

Instalación de SQL Server

Para la elaboración de este tema se ha seleccionado el administrador de base de datos Sybase Versión 11. La razón de esto es porque es común, existen algunas versiones de distribución libre y la forma de administrarlo es similar a otros administradores de base de datos. La empresa que lo distribuye es Adaptive Server Enterprise (ASE).

Antes de iniciar la instalación debemos realizar algunas actividades:

- Ø Definición de:
  - Ø Nombre del Servidor de SYBASE
  - Ø Nombre del equipo donde va a residir
  - Ø Tamaño del dispositivo maestro en Megabytes
  - Ø Definición por cual puerto va a correr Sybase
  - Ø Variables de ambiente: $DSQUERY(Nombre del servidor donde se encuentra la Base de Datos), $DSLISTEN(Nombre del servidor donde se encuentra la Base de Datos) y $SYBASE (Path donde se encuentra el ejecutable de Sybase)
  - Ø Definir donde se ubicará el archivo ERRORLOG. En este archivo se registrarán los errores de instalación. Se crea en el proceso de instalación.
- Ø Configuración de archivos:
  - Ø Configurar el archivo INTERFASES, que se encuentra en $SYBASE, con el número de puerto donde va a correr SYBASE. Este archivo proporciona detalles de conectividad para todos los servidores y se debe instalar en cada nodo.
  - Ø El archivo SYBASE.CFG con los parámetros adecuados.
- Ø Ejecución de:
  - Ø Crear y preparar la cuenta Sybase,
  - Ø Cargar los archivos Sybase (Sybsetup)
  - Ø Instalar srvbuild
  - Ø Inicializar dispositivos, crea base de datos, preparar cuenta de usuarios
  - Ø Cargar las Variables de Ambiente. $DSQUERY y $SYBASE en los equipos clientes. $DSLISTEN y $SYBASE en el servidor.
  - Ø Ejecutar el archivo RUNSERVER que es un Script para arrancar SYBASE.
  - Ø Ejecutar SP_CONFIGURE que cambia interactivamente los parámetros del archivo SYBASE.CFG creando un archivo de respaldo nombrado SYBASE.NNN, donde NNN es un número consecutivo de acuerdo a la versión del archivo SYBASE.CFG.

Con este administrador de base de datos existen dos versiones relativamente más importantes, estas son:

- Ø Adaptive Server. RDBM Normal
- Ø BackUp Server. Servidor de respaldo de un Adaptive Server, el cual como su nombre lo indica respaldara todo lo del servidor.

La diferencia en la configuración de estos servidores es mínima y se explicará en la sección de “Ejemplo de Configuración”.

Carga de los archivos de SyBase

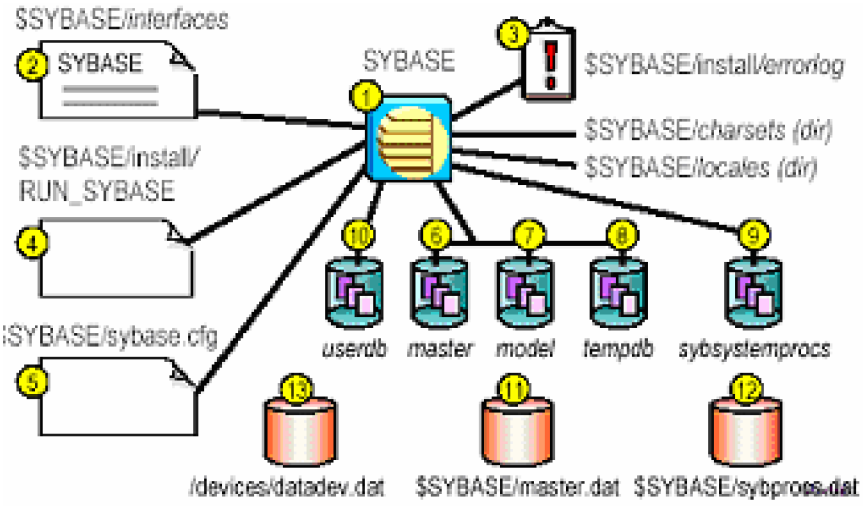
La carga se realiza como opción del disco de instalación. Después de la descarga queda la siguiente estructura de archivos:

SYBASE

Install	Instala programas archivos RUNSERVER files
Bin	Executables (Por ejemplo, isql, dataserver)
Include	Archivos de encabezado para librerias
Lib	Librerias del lenguaje del servidor

Config	Archivos de configuración de la librería de red
Scripts	Escripts SQL, Base de datos de ejemplo de Escripts
Sample	Código Ejemplo
Init	Archivos de instalación del log
Pad	Archivo de base dato para almacenamiento eficiente
Xappdefaults	Instalación y utilerias de los archivos de default
Sybhelp	Texto de los mensajes de ayuda
Charsets	Archivos de configularion de la librería de red
Locales	Set de carácteres y archivos de clasificación
Upgrade	Programas de Actualización
Diag	Herramientas de Diagnostico

El programa de instalación inicializa el dispositivo master y prepara las siguientes:
Ø Las bases de datos master, model y tempdb se instalan en el dispositivo master
Ø Dentro de la base de datos master las tablas: Sybsystemprocs, sybsecurity*
Ø La base de datos sybsystemprocs es instalado en el dispositivo que escoja
Ø Para cada base de datos, tres segmentos son creados: system, default y logsegment
Ø sybsecurity es instalada sobre su propio dispositivo



Ejemplo del Archivo RUNSERVER

```
El Formato es RUN_SERVERNAME
#!/bin/sh
# SQL Server Information:
# name: SYBASE
# master device:/work/sybase/ASE/devices/master.dat
# master device size: 10752
# errorlog: /work/sybase/ASE/install/errorlog
# interfaces: /work/ASE
#
/work/sybase/ASE/bin/dataserver -c/work/sybase/ASE/servername.cfg
  -d/work/sybase/ASE/devices/master.dat \ -sSYBASE \
  -e/work/sybase/ASE/install/errorlog \  -i/work/sybase/ASE
```

Ejemplo de Requisitos en la Preinstalación

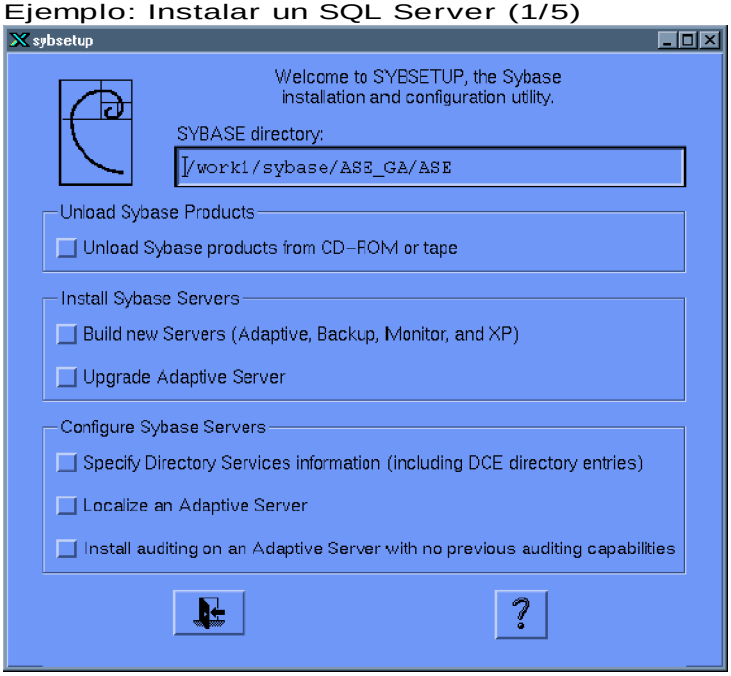
Parámetro	Usado en un ejemplo de srvbuild
Server name	SYBASE
Master device location	/work/ASE/devices/master.dat
Master device size	60 MB (y el DB master está en el dispositivo
Language	Us_English
Character set	ISO 8859-1
Sort order	Binary
Activate auditing (y/n)	No

Nombre de Dispositivo para	
sybsystemprocs database	/work/ASE/devices/sybprocs.dat
Error log location	/work/ASE/install/errorlog
Backup Server name	SYB_BACKUP

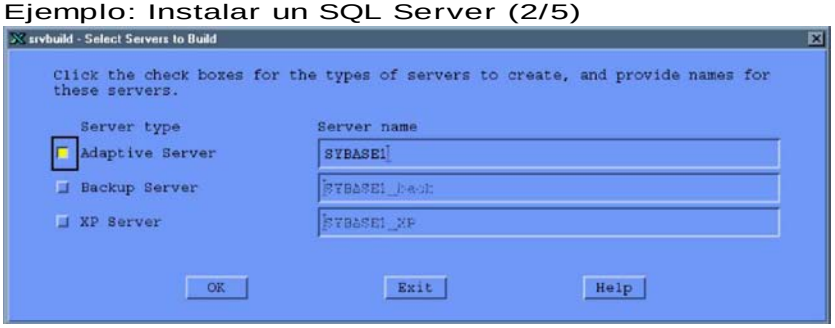
Recuerde !!
Ø Estar seguro que la cuenta sybase sea dueño del dispositivo master

Ø Arrancar un SQL Server como sybase y no como un súper usuario/root
Ø En el archivo interfaces, las líneas query y master deben comenzar con un tabulador

Ejemplo: Instalar un SQL Server (1/5)

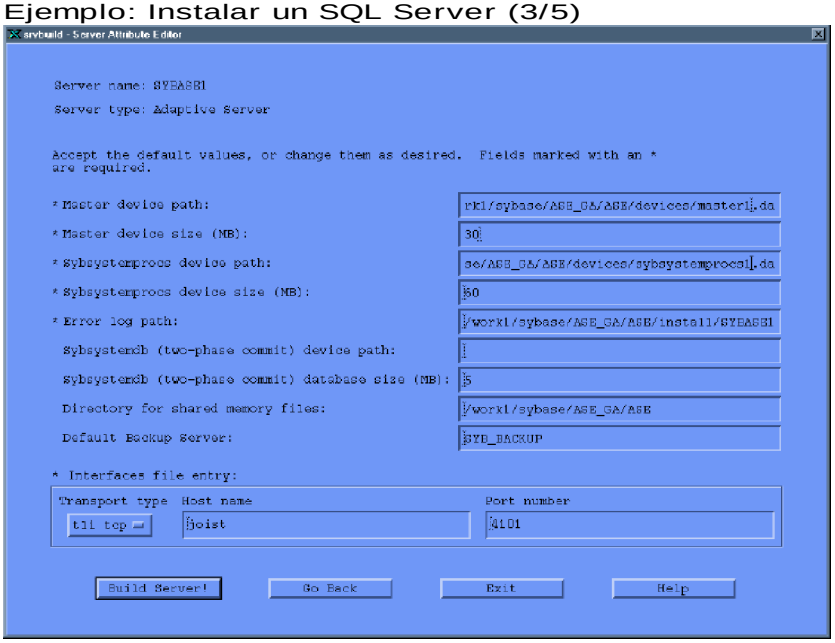


Ejemplo: Instalar un SQL Server (2/5)

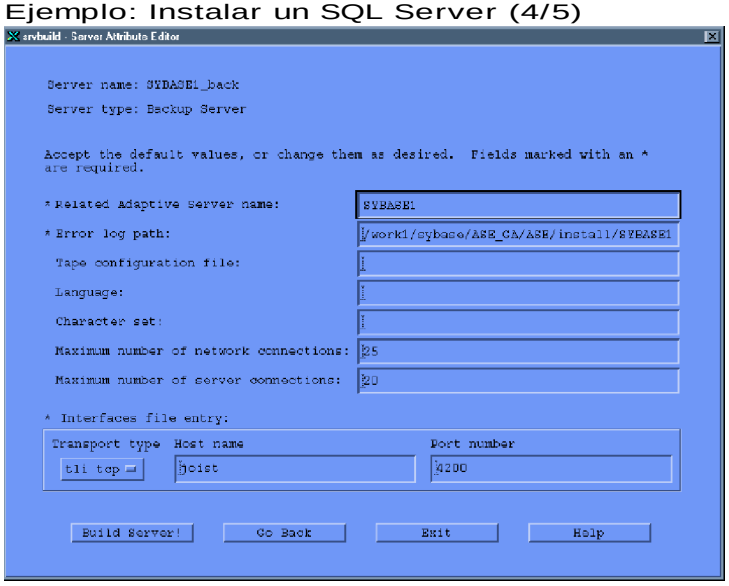


En esta pantalla se selecciona que tipo de servidor SQL se va a instalar Adaptative o BackUp. Si se selecciona Adaptive se pasa a la pantalla 3 en caso contrario a la 4. Después de la pantalla antes seleccionada se pasará a la pantalla 5.

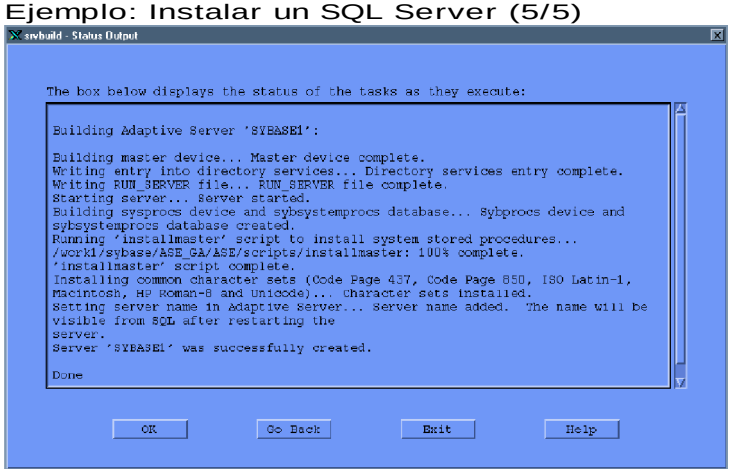
Ejemplo: Instalar un SQL Server (3/5)



Ejemplo: Instalar un SQL Server (4/5)



Ejemplo: Instalar un SQL Server (5/5)



Creando Servidores desde Archivos Fuente

Puede crear un SQL Server o un Backup Server con valores especificados en un archivo fuente que define los atributos del servidor. Para crear un servidor desde un archivo fuente, use el comando SRVBUILDRES

```
% srvbuildres -r
```

Ejemplo de un Archivo Fuente

```
srvbuild1112.001-SYBASE.rs
srvbuild.release_directory: /work1/sybase/ASE_GA/ASE
srvbuild.product: sqlsrv
srvbuild.server_name: SYBASE
srvbuild.new_config: yes
srvbuild.do_add_server: yes
srvbuild.do_upgrade: no
srvbuild.network_protocol_list: tcp
srvbuild.network_hostname_list: joist
srvbuild.network_port_list: 4100
srvbuild.master_device_physical_name: /work1/sybase/ASE_GA/ASE/
                                      devices/master.dat
srvbuild.master_device_size: 30
srvbuild.errorlog: /work1/sybase/ASE_GA/ASE/install/SYBASE.log
srvbuild.sybsystemprocs_device_physical_name:
                    /work1/sybase/ASE_GA/ASE/devices/sybsystemp rocs.dat
srvbuild.sybsystemprocs_device_size: 60
srvbuild.default_backup_server: SYB_BACKUP

srvbuild1112.002-SYBASE_back.rs
srvbuild.release_directory: /work1/sybase/ASE_GA/ASE
srvbuild.product: bsrv
srvbuild.server_name: SYBASE_back
srvbuild.do_add_backup_server: yes
srvbuild.network_protocol_list: tcp
```

```
srvbuild.network_hostname_list: joist
srvbuild.network_port_list: 4202
srvbuild.language: USE_DEFAULT
srvbuild.character_set: USE_DEFAULT
srvbuild.tape_config_file: USE_DEFAULT
srvbuild.errorlog: /work1/sybase/ASE_GA/ASE/install/SYBASE_back.log
```

Desconexión y Arrangue de un SQL Server

Use el comando shutdown para desconectar un servidor

```
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
Unexpected EOF from SQL Server.
```

Use el comando startserver para arrancar un servidor

```
% cd $SYBASE/install
% startserver -f RUN_SYBASE
[startup messages deleted]
```

showserver. Verfica que el servidor esté arriba y ejecutándose

```
   $ showserver
USER     PID %CPU %MEM  SZ  RSS TT STAT START  TIME CMND
sybase 12155  0.0  6.1 900 1672 pb    S 13:52  2:48
```

Configuración Predeterminada del SQL Server

Por qué Importa la Configuración?

- Ø Los valores de los parámetros de configuración determinan los recursos del sistema
- Ø La configuración del SQL Server afecta directamente su rendimiento
- Ø Los valores de configuración predeterminados, son mínimos y sólo permiten arrancar el SQL Server
- Ø Los administradores del sistema pueden reconfigurar muchos valores
- Ø Los oficiales de seguridad del sistema controlan:
  - Ø Expiración de contraseñas
  - Ø Actualización de las tablas del sistema
  - Ø Tamaño de los log de Auditoria

Que puede ser configurado en el SQL Server?

- Ø Respaldos y Recuperación
- Ø Administración del Cache
- Ø Disk I/O
- Ø Red
- Ø Recursos del Sistema Operativo
- Ø Memoria
- Ø Procesador
- Ø User environment
- Ø Lock manager
- Ø Parallel query

Opciones de Configuración Estáticas y Dinámicas

- Ø Los parámetros de configuración son estáticos o dinámicos
- Ø Los parámetros dinámicos afectan inmediatamente después de ejecutarse SP_CONFIGURE
- Ø Los parámetros estáticos requieren que SQL Server reasigne memoria. Al cambiar los parámetros estáticos se requiere reiniciar SQL Server

Archivo de Configuración

- Ø SQL usa un archivo de configuración que asigna los recursos del servidor
- Ø SQL Server utiliza $SYBASE/<SERVERNAME>.cfg a menos que se especifique otro archivo
- Ø SQL Server genera un nuevo archivo de configuración cada vez que un valor es cambiado usando SP_CONFIGURE

Ejemplo de un Archivo de Configuración

```
#Configuration File for the Sybase SQL Server
 [Configuration Options]
[General Information]
[Backup/Recovery]
   recovery interval in minutes = DEFAULT
[Cache Manager]
```

```
  procedure cache percent = DEFAULT
[Named Cache: default data cache]
  cache size = DEFAULT
[Meta-Data Caches]
  number of open databases = DEFAULT [Disk I/O]
[Network Communication]
  default network packet size = DEFAULT
[O/S Resources]
  max async i/os per engine = DEFAULT
[Parallel Query]
  number of worker processes = DEFAULT
[Physical Resources]
[Physical Memory]
  total memory = DEFAULT
[Processors]
  max online engines = DEFAULT
[SQL Server Administration]
  default database size = DEFAULT
[User Environment]
  number of user connections = DEFAULT
[Lock Manager]
  number of locks = DEFAULT
[Security Related]
  systemwide password expiration = DEFAULT
[Extended Stored Procedure]
  esp unload dll = DEFAULT
[Error Log]
  event logging = DEFAULT
[Rep Agent Thread Administration]
  enable rep agent threads = DEFAULT
[Component Integration Services]
  enable cis = DEFAULT
```

Problemas en el Manejo DBA

- Ø Checar permisos y protección del archivo de configuración
- Ø Borrar archivos inutilizables, para evitar que el sistema de archivos se llene
- Ø Especificar una bandera en la línea de comandos al DATASERVER (en el archivo RUNSERVER) para usar un archivo de configuración específico:

```
/work/sybase/ASE/bin/dataserver -d/devices/master.dat \ -sSYBASE
-e/work/sybase/ASE/install/errorlog \ c/work/sybase/ASE/SYBASE.010 &
```

Respaldo de un Archivo de Configuración

- Ø Al reiniciar, el servidor crea: $SYBASE/<SERVERNAME>.bak
- Ø Este archivo contiene los valores configurados hasta ese momento
  - Ø El archivo no es renombrado como archivo de configuración; es sobrescrito en cada arranque
- Ø Este archivo proporciona una copia de respaldo de el archivo de configuración en el evento en el que accidentalmente se sobrescribió

Cuando se Usan los Archivos de Configuración

- Ø Los archivos de configuración con diferentes configuraciones pueden ser utilizados para muchas situaciones
  - Ø Requerimientos específicos de procesamiento
  - Ø Estandarizar una configuración en la empresa
  - Ø Recuperación de una Base de Datos

Configurando SQL Server usando SP_CONFIGURE

- Ø Use el comando SP_CONFIGURE para reconfigurar SQL Server
- Ø Se puede reconfigurar SQL Server
  - Ø Interactivamente usando SP_CONFIGURE o
  - Ø Editando directamente el archivo de configuración
- Ø SP_CONFIGURE también para leer o escribir el archivo de configuración
- Ø Despliega parámetros de configuración en grupos jerárquicos
- Ø Muestra valores de configuración y de ejecución
- Ø Cambia los valores de configuración:

```
sp_configure "number of locks", 1000
```

Salida de SP_CONFIGURE

```
1> sp_configure
2> go
Group: Physical Memory
Parameter Name                 Default Memory Used Config Value Run Value
------------------------------ ------- ----------- ------------ ---------
```

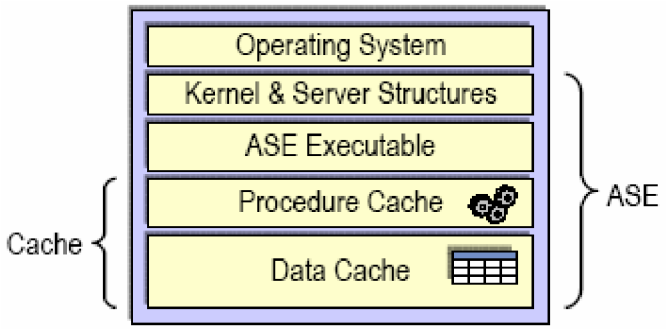
```
additional network memory          0            0            0          0
lock shared memory                 0            0            0          0
shared memory starting address     0            0            0          0
total memory                    7500            0         7500       7500

Group: Backup/Recovery
Parameter Name                 Default Memory Used Config Value Run Value
------------------------------ ------- ----------- ------------ ---------
Recovery interval in minutes         0           0            5         5
tape retention in days               0           0            0         0
recovery flags                       0           0            1         0

Group Name: Cache Manager
Parameter Name                 Default Memory Used Config Value Run Value
------------------------------ ------- ----------- ------------ ---------
total cache size                     0           0            0         0
size of data cache                   0           0            0         0
size procedure cache                 0           0            0         0
(coamtrips                           0           0            0         0
(cindextrips                         0           0            0         0

Group: SQL Server Administration
Parameter Name                 Default Memory Used Config Value Run Value
------------------------------ ------- ----------- ------------ ---------
allow nested triggers                1           0            1         1
allow updates to system tabl         0           0            0         0
audit queue size                   100           0          100       100
cpu accountin flush interval      l200           0          200       200
cpu grace time                     500           0          200       500
```

Revisión de la Sintaxis

- SP_CONFIGURE. Despliega los parámetros (dependiendo del nivel de desplegado del usuario)
- SP_CONFIGURE [group name]. Despliega parámetros de el grupo especificado
- SP_CONFIGURE [optname [, optvalue]]. Cambia un parámetro en el archivo de configuración y en sysconfigures (si es dinámico)
- SP_CONFIGURE "configuration file", [0, Read | verify | write | restore,"<filename>"] .
  - Sin parámetro despliega el archivo de configuración, en caso contrario permite acceso
- Los parámetros de el archivo de configuración son agrupados por función, haciendo más fácil su interpretación
- Se pueden preparar diferentes archivos de configuración para diferentes tipos de procesamiento
- La nueva opción del SP_CONFIGURE "configuration file", permite leer y escribir el archivo de configuración

Uso de Memoria del SQL Server

- El total de memoria física incluye:



Asignación de Memoria del SQL Server al Arrancar

SQL Server preasigna memoria en esta secuencia:

- Ejecutables del SQL Server
- Memoria Estática
- Memoria para parámetros de usuario
- Estructuras de datos sin caché
- La memoria que queda se divide entre:
  - Caché de datos
  - Cache de procedimientos (basado en el valor del parámetro de configuración procedure cache size configuration)

Maximizando la Memoria del SQL Server

- La memoria que está disponible son para buffers internos y cachés
- Teniendo una adecuada memoria disponible para cachés reduce el número de veces que el SQL Server tiene que leer datos del disco por información estática o planes de procedimientos compilados
- No hay fallas en el rendimiento por configurar el SQL Server para que utilice la memoria máxima disponible en la computadora. Sin embargo, asegúrese de evaluar otras necesidades de memoria en su sistema y estar seguro que SQL Server usa sólo la memoria disponible que queda
- SQL Server no reinicia si no puede tomar la memoria en la cuál esta configurado

Pasos para maximizar la memoria en el SQL Server

1. Determine la cantidad total de memoria física en su computadora
2. Reste la memoria requerida por el sistema operativo de la memoria física total
3. Si la máquina no es sólo para SQL Server, reste la memoria que es utilizada para otras aplicaciones.
   - Por ejemplo, reste la memoria que utilizan las aplicaciones cliente ejecutándose en una máquina SQL Server
   - Los sistemas de ventanas, como X Windows, requieren otro tanto de memoria y que puede interferir con el rendimiento del SQL Server
4. Reste la memoria que quiera asignar al parámetro de configuración additional network memory

Monitorear la Memoria Usando dbcc memusage

- La siguiente salida de dbcc es de una instalación normal en la cuál total memory es configurada a 12,000 páginas

```
1> dbcc memusage
2> go
Memory Usage:
                        Meg.  2K Blks     Bytes
Configured Memory:   23.4375    12000  24576000
Code size:            5.7627     2951   6042600
Kernel Structures:    2.4924     1277   2613502
Server Structures:    5.0214     2571   5265312
Cache Memory:         7.9961     4094   8384512
Proc Buffers:         0.0703       36     73728
Proc Headers:         2.0918     1071   2193408
```

Monitorear la Memoria Usando el Error Log

- Examine los mensajes de arranque en el error log

```
00:95/11/10 08:47:20.94
  server Number of proc buffers allocated: 543.
00:95/11/10 08:47:21.12
  server Number of blocks left for proc headers: 591.
00:95/11/10 08:47:21.12
 server Memory allocated for the default data cache cache: 4078 Kb
```

Fijando las Conexiones de Usuario

- La opción user connections fija el número máximo de conexiones de usuario que pueden conectarse al SQL Server
- El sistema operativo puede limitar el número total de conexiones
- Situación: Actualmente hay 25 usuarios; agregue conexiones de 15 usuarios más
  - Primero, planee cuánta memoria se requerirá; aproximadamente 95KB de memoria por espacio de trabajo por conexión de usuario (la cantidad es diferente para cada plataforma)
  - Después de que SQL Server ha sido reinicializado, la memoria para la conexión de usuario adicional será tomada de los cachés de datos y procedimientos

Asignación de Memoria después de Agregar 15 Usuarios

1. Reconfigure el número de usuarios

```
1> sp_configure  "user connections", 40
2> go
Parameter Name                 Default Memory Used Config Value Run Value
-----------------------------  ------- ----------- ------------ ---------
number of user connections          25        2125           40        25

 Configuration option changed. The SQL Server must be rebooted before the
change in effect since the option is static.
```

2. Cierre y reinicie SQL Server
3. Cheque la asignación de memoria ejecutando dbcc memusage

```
1> dbcc memusage
2> go
  Memory Usage:
                      Meg.   2K Blks      Bytes
Configured Memory:  23.4375     12000   24576000
Code size:           5.7627      2951    6042600
Kernel Structures:   3.0652      1570    3214066
Server Structures:   5.8215      2981    6104320
Cache Memory:        6.9062      3536    7241728
Proc Buffers:        0.0607        32      63648
Proc Headers:        1.8184       931    1906688
```

Diferencia

	Actual	Anterior	Actual - Anterior
Configured Memory	24576000	24576000	0
Code Size	6042600	6042600	0
Kernel Structures	3214066	2613502	600564
Server Structures	6104320	5265312	839008
Cache Memory	7241728	8384512	-1142784
Proc Buffers	63648	73728	-10080
Porc Headers	1906688	2193408	-286720

Recuperando el caché de datos y procedimientos perdido

Para recuperar los cachés de procedimientos y de datos, incremente la memoria total en 1,439,572 bytes (702 2K páginas)

1. Para recuperar el caché de datos y procedimientos, configure total memory agregando el número calculado de páginas perdidas del caché + 1.5% = (Memoria total actual + número de páginas para conexiones + sobrecarga)

12000 + 702 + (702 * 0.015) = 12712

```
1> sp_configure "total memory", 12712
2> go
00:00000:00007:1998/01/20 06:55:21.83 server Configuration file
'/work1/sybase/ASE_GA/ASE/SYBASE.cfg' has been written and the previous
version has been renamed to '/work1/sybase/ASE_GA/ASE/SYBASE.069'.
00:00000:00007:1998/01/20 06:55:21.93 server The configuration option
'total memory' has been changed by 'sa' from '12000' to '12712'.

Parameter Name           Default Memory Used Config Value Run Value
-------------- -------- ------- ----------- ------------ ---------
total memory               12000       24000        12712     12000

Configuration option changed. The SQL Server must be rebooted before the
change in effect since the option is static.
```

2. Cierre y reinicie SQL Server
3. Cheque la memoria asignada ejecutando dbcc memusage

```
1> dbcc memusage
2> go
  Memory Usage:
                      Meg.   2K Blks      Bytes
Configured Memory:  24.8281     12712   26034176
Code size:           5.7627      2951    6042600
Kernel Structures:   2.6616      1363    2790872
Server Structures:   5.4238      2777    5687248
Cache Memory:        8.6465      4427    9066496
Proc Buffers:        0.0760        39      79704
Proc Headers:        2.2539      1154    2363392
```

Otras opciones que Requieren Memoria

Ø Estas variables de configuración requieren memoria:
- Ø number of open databases (aproximadamente 35K cada una)
- Ø number of devices (aproximadamente 512 bytes cada una)
- Ø number of open objects (< 1000 bytes cada una)
- Ø number of locks (<100 bytes cada una)

Asignación de Recursos y Dispositivos

Recursos de disco

El administrador del sistema es el responsable de la administración de los recursos del equipo. Es el responsable de darle mantenimiento a los dispositivos de almacenamiento. Esto es, dar da alta, baja, modificarlos o asignar el uso de el.

El término dispositivo no se refiere necesariamente a un dispositivo físico diferente. Puede ser alguna partición de disco o un archivo en el sistema de archivos.

El termino partición se refiere a una asignación de almacenamiento que no puede alterarse.

Para hacer uso de un disco hay que inicializarlo, esto es prepararlo para que SYBASE lo reconozca.

El administrador también debe garantizar no perder la información. Un método puedes ser guardando el log de las bases de datos en un dispositivo físico diferente o duplicado de discos.

El administrador también es responsable que el equipo responda lo mejor posible por lo que se recomienda:

- Ø Colocando los objetos de base de datos y logs sobre dispositivos separados:
- Ø Colocando una tabla en un disco duro y los índices nonclustered sobre otro.
- Ø Distribuyendo las tablas grandes a través de diferentes discos puede mejorar el rendimiento, especialmente en las aplicaciones multiusuario
- Ø Ponga bases de datos con requerimientos de rendimiento crítico en dispositivos separados; ponga las bases de datos de servidor (master, tempdb) en dispositivos separados
- Ø Ponga tablas con uso pesado en discos separados
- Ø Use segmentos como necesidad para tablas críticas
- Ø Use particiones como necesidad para consultas paralelas
- Ø Dispositivos duplicados en discos separados
- Ø Las tablas particionadas equilibran el número de dispositivos físicos en un segmento

Segmentos

Introducción al Uso de Segmentos

- Ø Una base de datos puede estar sobre varios dispositivos
- Ø Se crean Segmentos de espacio en uno o más dispositivos lógicos para una base de datos
- Ø Cuando un segmento ha sido creado, se pueden poner tablas o índices en ese segmento

```
create table tableA(. . .) on seg1
```

Por Qué Usar Segmentos?

- Ø Use segmentos para controlar la colocación de datos de los objetos de base de datos y dispositivos base de datos
- Ø Controlar el uso del espacio:
  - Ø Use segmentos para limitar el tamaño de la tabla; una tabla no puede crecer más que su asignación en el segmento
  - Ø Tome ventaja del manejador Threshold para monitorear el uso del espacio

Segmentos Definidos por el Sistema

- Ø Cuando una base de datos es creada, el espacio asignado generan tres segmentos: system, default y logsegment, los cuales por default se ubican en el mismo segmento.
- Ø Ejemplo de base de datos: Datos y log en dispositivos separados

```
create database salesdb on data_dev1 = 5
      log on log_dev2 = 2
```

Agregando Segmentos Definidos por el Usuario

Para definir un segmento, ejecute SP_ADDSEGMENT

```
sp_addsegment segname, dbname, device_name

disk init name = "data_dev3",...
go
alter database salesdb on data_dev3 = 1
go
use salesdb
. go
sp_addsegment seg1, salesdb, data_dev3
go
```

Después de los comandos ALTER DATABASE y SP_ADDSEGMENT, data_dev3 tiene los segmentos system, default y seg1 de salesdb. El definir un segmento no asigna espacio adicional, etiqueta el espacio que ya ha sido asignado

Borrando Segmentos de Sistema y Predeterminados

Ejecute SP_DROPSEGMENT para borrar los segmentos system y default. Ejemplo:

```
sp_dropsegment "system", salesdb, data_dev3
sp_dropsegment "default", salesdb, data_dev3
```

Creando Objetos sobre Segmentos

Para colocar un objeto en un segmento, especifique el segmento cuando se crea el objeto. Si no especifica un nombre de segmento, la tabla es puesta en el segmento default. Asegúrese que los objetos que ponga en un segmento de usuario no compitan con las tablas del sistema y otros objetos. Los índices Clustered y sus tablas siempre están en el mismo segmento

```
create table tablename(..) on segname
create index indexname on tablename on segname
sp_placeobject segname, object
```

Desplegando Información acerca de los Segmentos

Ø Ejecute SP_HELPSEGMENT para listar los segmentos de la base de datos actual.

```
sp_helpsegment
```

```
Segment   name        status
      0   system           0
      1   default          1
      2   logsegment       0
      3   seg1             0
```

Ø Los nombres de segmentos son específicos en las base de datos
  Ø No pueden ser confundidos con nombres de segmentos iguales en otras bases de datos
  Ø Los Segmentos son listados en orden de su creación

Desplegando Información— Segmentos & Dispositivos

Ø Ejecute el siguiente comando para desplegar segmentos para esa base de datos

SP_HELPDB *databasename*

```
Name  db_size owner dbid  created     status
Sales    4 MB    sa    5  Oct 16 1992 no options set

device_fragments  size usage      free kbytes
----------------  -------------- -----------
data_dev1         2 MB data only         1376
log_dev2          1 MB log only          1008
data_dev3         1 MB data only         1008

device       segment
------------ ----------
data_dev1    default
data_dev1    system
log_dev2     logsegment
data_dev3    seg_1
```

Desplegando Información acerca de Objetos en Segmentos

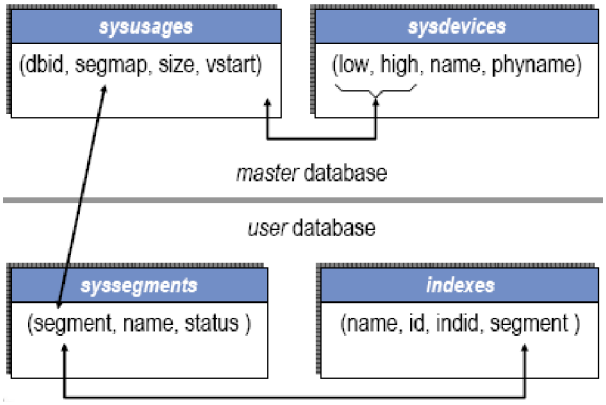
Ø Ejecute SP_HELPSEGMENT [segname] para listar objetos en el segmento y mostrar el dispositivo(s) relacionados. Ejemplo:

```
1> sp_helpsegment seg1
2> go
```

```
segment name status device     size free-pages
------------ ------ --------- ----- ----------
seg1              0 data_dev3 1.0MB         430

table_name index name indid
---------- ---------- -----
tableA     x_tableA       1
```

Tablas del Sistema relacionadas a Segmentos

Ø Dos tablas en la base de datos master: sysusages y sysdevices.
Ø Dos tablas en la base de datos de usuarios : syssegments y sysindexes
Ø Las tablas saben el lugar de las bases de datos, tablas e índices (incluyendo syslogs)
Ø La relación entre las tablas se da entre segmap en sysusuages y segment en syssegments



Que Hacer si hay otra Tabla Grande?

Ø Solución: Crear un nuevo segmento en un dispositivo nuevo. Comandos:

```
disk init; alter database; sp_addsegment;
sp_dropsegment (system and default); create table...on seg2
```

Si no hay Suficiente Espacio en Disco?

Ø Solución: Extienda seg1 a otro dispositivo. Comandos:

```
disk init; alter database;
sp_extendsegment segname, dbname, device_name
sp_dropsegment (system and default)
```

Cambiando Segmentos para Objetos Existentes

Ø Dirijase a las tablas o índices que crecen para ejecutar: SP_PLACEOBJECT segname, object
Ø Cuando la tabla A necesita más espacio, SQL Server utiliza seg2
Ø Divida una tabla grande hacia dos segmentos o dispositivos

Particiones

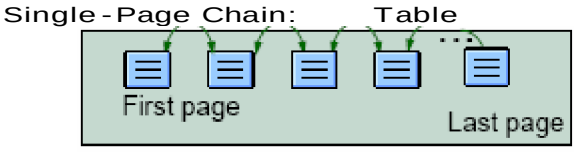
Usando Particiones

Ø Al particionar una tabla se crean cadenas multipágina para una tabla
Ø Al particionar se agrega un grado de paralelismo a los sistemas configurados para realizar procesamientos de consulta paralelo
Ø Cada proceso lee una partición por separado
Ø El particionar hace posible llenar una tabla en paralelo con BULK COPY
Ø El particionar hace posible distribuir el I/O de una tabla a través de múltiples dispositivos de base de datos
Ø El particionar proporciona muchos puntos de inserción para varias tablas

Un diseñador de base de datos escoge una o más tablas que pueden tomar ventaja de la partición de datos en una base de datos

Ø Un candidato es una tabla “append only” en la que en cada transacción debe escribir
  Ø Las tablas que proporcionan una historia o lista de auditoría son también candidatas para la partición de datos. Esto causa una alta disputa en la tabla

Inserciones hacia una Single- Page Chain Table

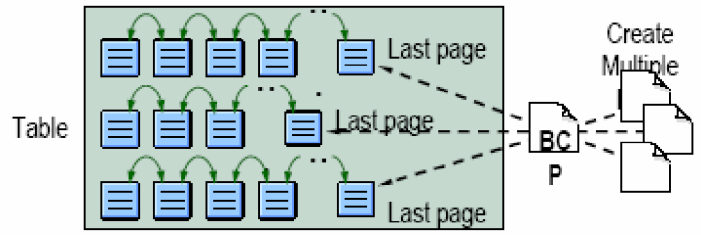


Ø Los objetos no particionados son implementados como una partición sencilla Todos los nuevos registros van a la última página
Ø Si la última página esta llena, una nueva página es asignada y los nuevos registros van a esta nueva página
Ø Una transacción de inserción mantiene un bloqueo exclusivo en la última página hasta el fin de la transacción

Inserts con Datos Particionados

Ø Dividir la tabla en múltiples particiones
Ø Cada inserción es realizada en una de las cadenas de página
  Ø Usted puede asignar las particiones en las que se hacen inserciones, con BCP o el sistema los seleccionará al azar

- Todas las inserciones de una transacción dada van hacia la misma partición



Inserciones usando Bulk Copy Paralelo

- Las tablas que son particionadas tienen mejor rendimiento cuando las inserciones son hechas mediante un bulk copy paralelo (BCP)
- Las inserciones son asignadas a particiones específicas usando BCP y ejecutadas concurrentemente

Tablas Particionadas y Procesamiento de Consulta Paralelo

- Si las tablas son particionadas, el procesamiento de consulta paralelo puede potencialmente producir mejoras en la realización de consultas
- Las particiones incrementan el número de cadenas de páginas que son accesadas simultáneamente por procesos
- Cuando las particiones son distribuidas a través de discos físicos, la disputa reducida de I/O acelera la velocidad de el procesamiento de consulta paralelo y logra un alto nivel de paralelismo
- El optimizador puede escoger usar el procesamiento de consulta paralelo para una consulta contra una tabla particionada cuando el procesamiento de consulta paralelo es habilitado

Equilibrio de Partición / Partición Inclinada

- Una distribución equitativa de páginas a través de particiones (llamada equilibrio de partición) incrementa la efectividad en búsquedas paralelas
- A veces los datos en una tabla particionada pueden llegar a ser distribuidas en forma irregular o inclinadas dado lo siguiente:
  - Un gran número de registros son insertados en un subconjunto de una tabla particionada
  - Un gran número de registros son borrados de un subconjunto de una tabla particionada
  - Las actualizaciones ocurren sobre un subconjunto de una tabla particionada usando algún otro modo de actualización
  - BCP copia un gran número de registros hacia un subconjunto de una tabla particionada o inserta múltiples archivos aleatoriamente

Distribuyendo Datos a través de Particiones con Índices Clustered

Por definición una tabla y su índice agrupado están en el mismo segmento. Si crea una tabla en un segmento y su índice agrupado en otro, la tabla se trasladará con su índice.

Particionando una Tabla sobre un dispositivo sencillo

- No puede crear una tabla como particionada. Primero debe crear la tabla, entonces use alter table para particionarla

1. Crear la tabla
   ```
   create table salesdetail (ord_no char(10), prod_no char(25),    )
   ```
2. Especificar el número de particiones para la tabla
   ```
   alter table salesdetail partition 4
   ```

Particionando a través de Múltiples Dispositivos de Base de Datos

1. Establecer un segmento que tenga múltiples fragmentos de datos:
   ```
   sp_addsegment seg1, salesdb, data_dev1
   sp_extendsegment seg1, salesdb, data_dev2
   ```
2. Poner la tabla en el segmento:
   ```
   create table salesdetail (column1 numeric(3,0), column2 char(25))on seg1
   ```
   - O use sp_placeobject después de crear un objeto:
     ```
     sp_placeobject seg1, salesdetail
     ```
3. Especifique el número de particiones para la tabla:
   ```
   alter table salesdetail partition 4
   ```

Particionando a través de Múltiples Dispositivos

```
disk init NAME='datadev1',PHYSNAME='/curdev1/server11/datadev1.dat',
VDEVNO=5, SIZE=1024
```

```
 disk init NAME='logdev1', PHYSNAME='/curdev1/server11/logdev1.dat',
VDEVNO=6, SIZE=512
 disk init NAME='datadev2', PHYSNAME='/curdev1/server11/datadev2.dat',
VDEVNO=7, SIZE=512
 disk init NAME='datadev3', PHYSNAME='/curdev1/server11/datadev3.dat',
VDEVNO=8, SIZE=512
 disk init NAME='datadev4', PHYSNAME='/curdev1/server11/datadev4.dat',
VDEVNO=9, SIZE=512
 create database testdb on datadev1=2, datadev2=1, datadev3=1, datadev4=1
 log on logdev1=1
 use testdb
 exec sp_addsegment 'tableseg','testdb','datadev2'
 exec sp_extendsegment 'tableseg','testdb','datadev3'
 exec sp_extendsegment 'tableseg','testdb','datadev4'
 exec sp_dropsegment 'system','testdb','datadev2'
 exec sp_dropsegment 'system','testdb','datadev3'
 exec sp_dropsegment 'system','testdb','datadev4'
 exec sp_dropsegment 'default','testdb','datadev2'
 exec sp_dropsegment 'default','testdb','datadev3'
 exec sp_dropsegment 'default','testdb','datadev4'
 create table a(column1 int, column2 char(30)) on tableseg
 alter table a partition 3
```

Restricciones en Particiones

- Las tablas que no pueden ser particionadas
  - Todas las tablas del sistema en la base de datos master
  - Tablas de trabajo
  - Tablas temporales
  - Tablas que ya están particionadas
- Operaciones que no están permitidas sobre tablas particionadas
  - truncate table
  - sp_placeobject
  - alter table .. partition

Otros Comandos sobre Objetos Particionados

- Para desparticionar una tabla particionada:

```
 alter table table_name unpartition
```

- Borrar una tabla particionada :

```
      drop table table_name
```

- Para usar particiones, use la sintaxis normal de insert, update, delete y select
  - La disputa entre un Insert y un update es reducido
  - Use sp_helparition para obtener información acerca de las tablas con múltiples particiones

Notas para el Uso de Particiones

- Usualmente sólo un pequeño número de tablas en una aplicación se benefician de la partición
- El mejoramiento del rendimiento puede ser significativo
- Use particiones de datos cuando se tenga una tabla con muchas inserciones
- Si hay una pequeña disputa sobre una tabla, el particionarla afectará mínimamente el rendimiento

Dispositivos

Inicializando dispositivos de base de datos

El SA inicializa un nuevo dispositivo de base de datos con el comando DISK INIT. DISK INIT mapea un disco físico (o un archivo del sistema operativo) a un nombre de dispositivo de base de datos. El nuevo dispositivo es insertado en master..sysdevices. Después de inicializado el dispositivo, se pueden crear base de datos en él. Use esta sintaxis:

```
   disk init name = "device_name", physname = "physical_name",
   vdevno = virtual_device_number, size = number_of_pages
```

Solo los administradores del sistema pueden usar DISK INIT. Antes de ejecutarlo:

- Respalde la base de datos master
- Asegúrese de que hay suficiente espacio en disco
- Asegúrese de que el archivo o dispositivo no haya sido ya inicializado
- Asegúrese que la cuenta sybase tenga permisos de escritura sobre ese dispositivo
- Máximo de dispositivos de base de datos configurables: 255
- Después de ejecutar DISK INIT, respalde la base de datos master

Dispositivos Predeterminados

- Con CREATE DATABASE, se especifica el dispositivo sobre el cuál la base de datos deberá ser creada. Si no hay un dispositivo especificado, la base de datos se crea sobre el dispositivo predeterminado

Ø Después de que un dispositivo es inicializado, el SA puede preparar el dispositivo como un dispositivo predeterminado usando SP_DISKDEFAULT

sp_diskdefault device_name, {defaulton | defaultoff}

Ø Ejemplo: Inicialice data_dev1 y entonces se agrega como un dispositivo predeterminado:

```
disk init name = "data_dev1", physname = "/dev/c0t1d2s3", ...
exec sp_diskdefault data_dev1, defaulton
```

Ø SP_DISKDEFAULT marca esos dispositivos como dispositivos predeterminados en sysdevices

Ø Cuando un SQL Server es instalado, el dispositivo master es un dispositivo predeterminado

Ø Recomendado: cambiar esto para master y así evitar desorden en la base. Ejecutar los siguientes comandos:

```
exec sp_diskdefault master, defaultoff
exec sp_diskdefault data_dev1, defaulton
```

Ø Con comandos SP_DISKDEFAULT por separado, se pueden especificar más de un dispositivo predeterminado

Ø Varios dispositivos predeterminados se van usando conforme un orden alfabético

Eliminando Dispositivos

Ø Al cambiar el tamaño del dispositivo, se debe quitar y luego volver a inicializar

Ø Para quitar un dispositivo:

sp_dropdevice device_name

Ø Entonces resetear el SQL Server para que pueda reusarse el número de dispositivo (vdevno)

Ø Para archivos de disco, borrar el archivo después de eliminar el dispositivo para liberar el espacio en disco

Ø No se puede borrar un dispositivo si aún contiene base de datos

Obteniendo información del dispositivo

Ø sysdevices lista base de datos y dispositivos de respaldo. Ejemplo de la salida de select * From sysdevices:

```
     Low      high status cntrltype name       physname       mirrorname
-------- -------- ------ --------- ---------- -------------- ----------
16777216 16782335      2         0 data_dev1  /dev/c0t1d2s3  NULL
       0    10239      3         0 master     d_master       NULL
       0    20000     16         3 tapedump1  /dev/rmt4      NULL
       0    20000     16         2 tapedump2  /dev/rst0      NULL
```

El procedimiento SP_HELPDEVICE consulta sysdevices, evalúa ciertos valores, e imprime información útil acerca de los dispositivos, como se muestra aquí:

```
Device Name Physic Name         description
----------- ------------------- ----------------------------------------
data_dev1   /dev/rsd2d          Special physical disk 10.00 MB
master      d_master            special default disk physical disk 20 MB
tapedump1   /dev/rmt4           tape 625 MB dump device
data_dev1   /dev/data_dev1.dat  special, physical disk, 10.00 MB
log_dev1    /dev/log_dev1.dat   special, MIRROR mirror='/dev/logev1_mira

status                    cntrltype Device Number low      high
------------------------- --------- ------------- --------- ---------
     2                            0             1 16777216  16782335
     3                            0             0        0    102390
    16                           30             0    20000
     2                            0             1 16777216  16782335
mirriored physical disk         733             0 33554432  33559551
```

Duplicado

Duplicado

Ø El espejo de un SQL Server duplica todo el contenido de un dispositivo

Ø Escribe en ambos dispositivos

Ø Si un disco de falla, SQL Server lo nota al tratar de leer o escribir en ese disco y por lo tanto, continua con el otro

Ø Beneficios

Ø Previene la interrupción de tareas y/o procesos debido alguna falla en el disco

Ø Asegura la recuperación total e ininterrumpida

Ø Costos

Ø Usa recursos adicionales (almacenamiento en disco)

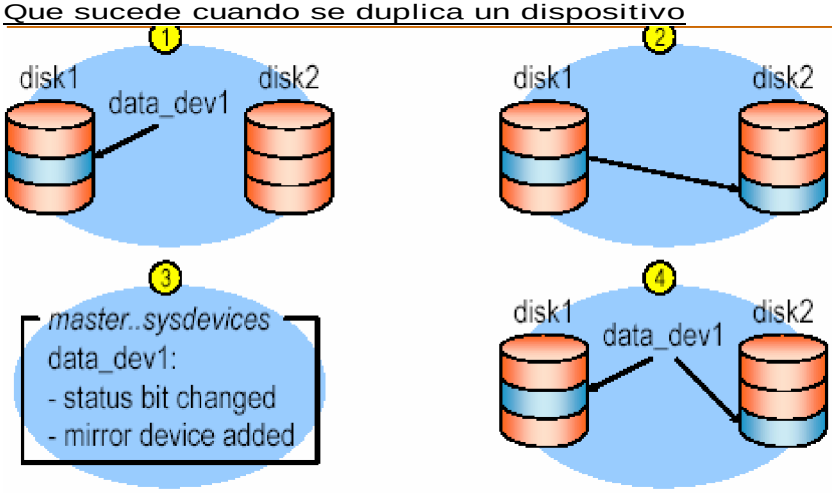
Ø Las escrituras son más lentas porque se están duplicando

Ø No obstante los costos, la duplicación de discos es recomendada ampliamente

¿Que se debe duplicar?

Ø Duplicar los dispositivos más valiosos y vulnerables
  Ø Master device
  Ø Log devices
Ø Recuerde poner el duplicado del log de transacciones sobre un dispositivo separado

Que sucede cuando se duplica un dispositivo



Comandos de duplicación de discos

Ø DISK MIRROR comienza el duplicado de discos
  Ø No inicialice el dispositivo duplicado con DISK INIT; un dispositivo de base de datos y su duplicado constituyen un solo dispositivo lógico.
  Ø El comando DISK MIRROR agrega el nombre del duplicado en la columna mirror name de la tabla sysdevices
Ø DISK UNMIRROR detiene el proceso de duplicado cuando es necesario el mantenimiento del hardware o el hardware del dispositivo necesita ser cambiado
Ø DISK REMIRROR restaura el proceso de duplicado que había sido suspendido debido a una falla del dispositivo o por el uso de DISK UNMIRROR

Desactivando el duplicado

Ø Intencional o Automático
  Ø Intencional: Ejecutando DISK UNMIRROR
  Ø Automático: Si un error I/O es detectado sobre algún disco
Ø ¿Que sucede si hay un error de I/O?
  Ø Los bits de estados están en sysdevices y ahí se indica la desactivación y cuál disco sigue operando
  Ø La I/O es deshabilitada en el dispositivo defectuoso
  Ø Se realiza un CHECKPOINT sobre la base de datos master
  Ø Un mensaje de error es enviado al error log
  Ø Cualquier proceso WAITFOR MIRROREXIT es habilitado

Resumen de los comandos de duplicación

Ø Duplicar un dispositivo:
```
disk mirror name = "device_name", mirror = "physical_name",
[ ,writes = {serial | noserial} ]
```
Ø Suspender el duplicado de un dispositivo:
```
disk unmirror name ="device_name"
```
Ø Continuar el duplicado de un dispositivo :
```
disk remirror name ="device_name"
```
Ø Deshabilitar el duplicado de un dispositivo permanentemente:
```
disk unmirror name ="device_name", mode = remove
```
Ø Deshabilitar el dispositivo primario temporalmente:
```
disk unmirror name ="device_name", side = "primary", mode = retain
```
Ø Para reinstalar el dispositivo primario, use disk remirror
Ø Desplegado de dispositivos duplicados
```
sp_helpdevice
```

Proporcionando el nombre del duplicado del dispositivo master en el arranque

Ø Si el dispositivo master es duplicado, edite el archivo RUNSERVER para agregar el nombre físico del duplicado

Ø Si hay un problema con el dispositivo primario master, SQL puede accesar con el duplicado
dataserver -ddevicename ...
[-rmastermirror_devicename]...

Administración de las Bases de Datos

Creación de Bases de Datos

Prerrequisitos al crear y darle tamaño a la base de datos

Ø Antes de crear base de datos, decida:
  Ø El tamaño de la base de datos
  Ø La ubicación y el espacio necesario
  Ø Si un dispositivo de log es necesario de que tamaño
Ø Se debe estar en la base de datos master para ejecutar el comando CREATE DATABASE
Ø El tamaño es en megabytes, mínimo 2MB
Ø El log es fácil de extender, imposible encoger
  Ø Para hacer más chica la base de datos , se debe volverla a crear y copiar datos de nuevo con la utilidad de copiado BCP
Ø Al estimar el tamaño de una base de datos, considere principalmente: Tablas, Índices y Log de transacciones
Ø Deje algún espacio libre, dependiendo de la actividad anticipada
Ø Use SP_ESTSPACE para estimar el tamaño de las tablas y sus índices

```
sp_estspace titles, 10000
```

```
name        type          idx_level Pages Kbytes
---------- ------------ --------- ----- ------
Titles      data                  0   983   1966
Titleidind clustered             0     7     14
Titleidind clustered             1     1      2
Titleind   nonclustered          0   279    558
Titleind   nonclustered          1     9     18
Titleind   nonclustered          2     1      2

Total_Mbytes: 2.50
(followed by summary data for titleidind, titleind)
```

Tamaño del log

Ø El tamaño del Log depende de la actividad (tipo y cantidad de transacciones) y la frecuencia de los respaldos
  Ø Un buen punto de partida: 10-25% del tamaño global de la base de datos
  Ø Todos los inserts, deletes y updates son registrados
Ø Simule correr aplicaciones con el mismo número de usuarios; entonces mida el uso del log usando DBCC CHECKTABLE (syslogs)

Recomendaciones en la asignación de espacios

Ø Coloque el log en un dispositivo diferente
  Ø Deje sus respaldos del log de transacciones en separado
  Ø Permita el duplicado de disco del log para un recuperación a tiempo

create database: Sintaxis

```
create database database_name
[on {default | database_device} [= size]
   [,database_device [= size]]...]
[log on database_device [= size,...]
   [,database_device [= size]]...]
[with override]
[for load]
```

Ø Ejemplos:

```
 (1) create database pubs2 on default = 4
 (2) create database mydb on default = 3, data_dev1 = 2
 (3) create database salesdb on sales_dev1 = 2 log on sales_dev1 = 2 with
override
```

Colocación de Datos

Ø En un pequeño sistema o en un entorno de desarrollo, se pone toda una base de datos en el mismo disco físico, especialmente si es fácil de volver a crear

- Ventajas:
  - Simple
  - Usa recursos mínimos
- Desventajas:
  - No hay una recuperación completa en caso de fallo en el medio (Solo el último respaldo)
  - El rendimiento se afecta si las tablas son grandes y la actividad es pesada
- Con tablas grandes, por las cuales la recuperación y el rendimiento son críticos:
- Poner base de datos en discos separados
- Poner tablas grandes en discos separados usando segmentos

Colocando Logs

En sistemas pequeños o en un entorno de desarrollo, almacenar el log y los datos en un mismo disco físico

- Ventajas:
  - Simple
  - Usa recursos mínimos
- Desventajas:
  - No se pueden realizar respaldos de log a menos que la base sea creada con la opción with override
  - En caso de un fallo en el medio, no se puede recuperar hasta el último minuto; solamente es bueno en el último respaldo completo
  - Con gran actividad, el rendimiento baja, porque las escrituras de el log compite con las escrituras de datos

Con grandes tablas y cuando la recuperación y rendimiento son críticos, las opciones siguientes son posibles:

- Ponga el log en un disco físico separado de los datos
- Ponga el log en un disco que no este ocupado
- Ponga el log en su propio segmento en el que se pueda usar el threshold manager para monitorear su uso
- Duplicar del dispositivo del log para una completa recuperación

Colocando Respaldos, sybsecurity

- Los respaldos son generalmente puestos en archivos de disco o cintas
- La base de datos sybsecurity contiene la tabla sysaudits
- Si espera una gran actividad auditable, ponga sybsecurity en:
  - En un disco que no este ocupado
  - En su propio segmento

¿Propiedad de la Base de Datos, Quién crea las bases de datos?

- Los administradores del sistema (logines con el rol SA) pueden crear bases de datos
- El login que crea una base de datos es propietario de ella inicialmente
- Para transferir la propiedad de la base de datos, accese a esa base da datos usando use DBNAME seguido de SP_CHANGEDBOWNER.
- Sintaxis: SP_CHANGEDBOWNER login_name
  - El login debe ser válido en el servidor. Ejemplo:

```
use productsdb
sp_changedbowner fred
```

- Nota: La propiedad de la base de datos master no puede transferirse

Otorgando permisos para create database. Los administradores del sistema pueden otorgar permisos de create database a usuarios específicos:

```
grant create database to mary
```

Efectos sobre las tablas del sistema

- Cuando se crea una base de datos, las siguientes tablas del sistema en master son afectadas:
  - Sysdatabases. Contiene un registro por cada base de datos en el SQL Server, incluyendo el nombre de la base de datos y su propietario
  - Sysusages. Contiene un registro para cada fragmento del dispositivo para cada base de datos, indicando el tamaño y la dirección del disco del comienzo lógico para ese fragmento
- Para desplegar esta información, consulte estas tablas o ejecute SP_HELPDB

```
sp_helpdb
name    db_size  owner  dbid  status
------- -------  -----  ----  --------------------
master   8.0 MB  sa        1  no options set
```

```
model     2.0 MB  sa          3  no options set
salesdb   6.0 MB  sa          5  select into/bulkcopy
tempdb    2.5 MB  sa          2  select into/bulkcopy
```

Cuando es seguido por un nombre de base de datos, SP_HELPDB reporta información sobre esa base de datos

SQL Server sabe en que dispositivo esta una base de datos a través de la relación de las tablas del sistema SysDatabase, SysUsages y SysDevices.

```
 select * from sysusages where dbid = db_id("smalldb")
dbid  Segmap  Lstart  Size  Vstart    Pad   unreservedpgs
----  ------  ------  ----  --------  ----  -------------
   5       3       0  1024  16777216  Null            680
   5       4    1024   512  33554432  Null            496
```

Para desplegar el mapeo de uno de estos fragmentos de dispositivos:

```
 select low, high, name, phyname, mirrowname from sysdevices where
16777216 between low and high
Low       High      Name       Physname             Mirrorname
--------  --------  ---------  -------------------  ----------
16777216  16782335  data_dev1  /syb/date_dev1.dat   NULL
```

SP_HELPDB db_name despliega ubicación y uso del disco para esa base de datos

```
name     db_size  owner  dbid  created      status
-------  -------  -----  ----  -----------  --------------
smalldb   3.0MB   sa        5  May 5, 1993 no options set

Device_Fragments  Size    Usage       Free Kbytes
----------------  ------  ---------   -----------
data_dev1          2.0 MB  data Only          1376
log_dev1           1.0 MB  data Only          1008
```

Personalizando base de datos. Cuando se crea una base de datos, el contenido de model es copiado a la nueva base de datos. Puede personalizar model para que contenga procedimientos almacenados, tablas, reglas, tipos de datos de usuario, privilegios y opciones para todas las futuras bases de datos. Solo el administrador del sistema puede actualizar model

Fijando las opciones de base de datos

Usando SP_DBOPTION, las siguientes opciones de base de datos pueden ser fijadas:

- Ø abort tran on log full
- Ø allow nulls by default
- Ø dbo use only
- Ø ddl in tran
- Ø identity in nonunique index
- Ø no chkpt on recovery
- Ø no free space accounting
- Ø read only
- Ø select into/bulkcopy/pllsort
- Ø single user
- Ø trunc log on chkpt

SSO pueden habilitar o deshabilitar free-space accounting; los demás serán fijados por el dueño de la base de datos (o un SA actuando como un dbo)

Las opciones no podrán ser cambiadas en la base de datos master

```
 sp_dboption [dbname, option_name, {true|false}]

 use master /* debe estar en master */
 go
 sp_dboption "smalldb", "read only", true
 go
 use smalldb /* debe usar la base de datos. */
 go
 checkpoint /* hacer checkpoint a la base de datos */
 go
```

- Ø Cualquier usuario puede usar SP_HELPDB para desplegar las opciones actuales de la base de datos

Monitoreando el uso del espacio

- Ø La base de datos, tanto el segmento de log como el de datos, pueden llenarse tanto como uso se le de a la base de datos

- Ø Si esto sucede, todas las modificaciones de datos son suspendidas
- Ø Puede usar las siguientes herramientas para monitorear el uso del espacio en la base de datos: SP_HELPDB, SP_HELPSEGMENT, SP_SPACEUSED. THRESHOLD MANAGER

Que hacer cuando no se tiene espacio

- Ø Si no se tiene espacio en el log, se tiene que truncarlo
  - Ø Posibles soluciones: Dump/truncate frecuentemente
- Ø Si no se tiene espacio en los segmentos de datos, intente liberar espacio eliminando objetos no usados
- Ø Otra alternativa: Extender la base de datos (datos y/o log)

Extendiendo una base de datos

Usando ALTER DATABASE, los propietarios de base de datos y administradores del sistema asignan espacio adicional a una base de datos sobre el mismo o diferentes dispositivos. Sintaxis:

```
alter database database_name
 [on {default | database_device} [= size]]
 [log on database_device [= size]]
```

- Ø El tamaño es especificado como incrementos de espacio adicional; predeterminado, 1MB

```
create database salesdb on data_dev1 = 5
alter database salesdb on data_dev2 = 2
alter database pubs2 on default = 2
```

Tips para la expansión de una base de datos

- Ø Puede expandir una base de datos mientras este en uso
- Ø Solo se puede expandir la base de datos master en el dispositivo master
- Ø Si tempdb o model es expandido y el dispositivo master es reconstruido entonces se vuelve a expandir
  - Ø No haga a model más grande que tempdb
  - Ø Si model es más grande que tempdb, SQL Server no reiniciará

Separando el log y moverlo a un dispositivo diferente

- Ø Para una base de datos creada sin la opción log on (esto es, sin un log separado), haga lo siguiente:
  - Ø Respalde el log al truncarlo
  - Ø Modifique la base de datos hacia un nuevo dispositivo
  - Ø Ejecute SP_LOGDEVICE para hacer ese dispositivo en dispositivo de log
- Ø Realice estos pasos con un mínimo de tiempo entre cada uno para evitar que los registros sean escritos al dispositivo original
- Ø Efecto:
  - Ø Las paginas antiguas serán asignadas al dispositivo anterior
  - Ø Los registros existentes serán desasignados tanto el log sea truncado
  - Ø Resultado: El log crece en el nuevo dispositivo

Eliminando base de datos

- Ø Los DBO y SAs pueden borrar una base de datos ejecutando DROP DATABASE
- Ø La base de datos no debe estar en uso
- Ø Cuando borra una base de datos se libera el espacio

Control de Acceso

Dentro del Sistema Manejador de Base de Datos (DBMS) podemos encontrar un acceso multicapas:

- Ø El usuario final debe tener una cuenta válida dentro de la capa del servidor (DBMS).
- Ø El usuario final debe ser un usuario válido dentro de la capa de la base de datos.
- Ø El usuario final deberá tener permiso dentro de la capa de los datos.

Usuarios en una Base de Datos

- Ø El programador de aplicaciones, quien se encarga de escribir los programas de aplicación que utilizan la base de datos.
- Ø El usuario final, el cual tiene acceso a los datos de la base a través de alguna aplicación desarrollada o utilizando una interfaz incluida como parte integral de los programas del DBMS.

Los roles o perfiles sirven como medio para conceder privilegios sobre todo el sistema a un usuario que los requiera. Estos permisos pueden verse reflejados sobre objetos o sobre el

mismo sistema. En los RDBMS ya vienen predeterminados algunos, sin embargo no en todos se pueden crear nuevos roles.

- Ø DBA. (database administrator, administrador de la base de datos).
- Ø DBO. (database owner, Dueño de la base de datos).
- Ø SSO. (Security System Oficer, Oficial del sistema de seguridad).
- Ø Oper. (Operador)

El Login sa

- Ø Cuando SQL Server en instalado por primera vez, el login sa:
  - Ø Ha sido asignado con los tres roles especiales (system administrator, system security officer y system operator)
  - Ø Ejecuta todos los comandos SQL
  - Ø Es propietario de la base de datos master
  - Ø Es tratado como dueño de la base de datos (dbo) en todas las bases de datos
  - Ø Tiene acceso a todas las bases de datos y objetos de base de datos
- Ø El password del login sa es inicialmente nulo, pero debe ser cambiado
  - Ø Una vez cambiado, no puede ser nulo otra vez
- Ø En la instalación, el login sa puede hacer todo
  - Ø Mantener esos tres roles administrativos
  - Ø Otorgarlos a uno o más logines y luego bloquear el login sa
  - Ø Revocar los roles del login sa

DBA. (Data Base Administrador, Administrador de la base de datos)

Lleva a cabo tareas administrativas inconexas a aplicaciones específicas. El administrador del sistema no es necesariamente único; el rol puede ser otorgado a cualquier número de cuentas individualmente y para ello las funciones del DBA deben estar centralizadas o muy bien coordinadas. Las tareas del DBA incluyen:

- Ø Decidir el contenido de la base de datos.
- Ø Crear la estructura de almacenamiento y los métodos de acceso.
- Ø Administrar y controlar la seguridad física y lógica de la base de datos.
- Ø Monitoreo del comportamiento y crecimiento de la base de datos.
- Ø Procedimientos de respaldo y depuración de la base de datos.
- Ø Salvaguardar la documentación, respaldos y diccionario de datos tanto de la base datos como de sus aplicaciones.
- Ø Procedimientos de contingencia y recuperación de la base de datos.
- Ø Modificar la base de datos o la descripción de la organización física.
- Ø Otorgar permisos de acceso y prioridades a los diferentes usuarios.
- Ø Especificar las limitaciones de integridad.
- Ø Ser el enlace con el usuario.
- Ø Instalación del servidor SQL.
- Ø Diagnóstico de problemas del sistema, así como la corrección de los mismos.
- Ø Otorgar y revocar roles en el servidor.
- Ø Configuración del servidor SQL.
- Ø Control de procesos en el servidor.

DBO. (DataBase Owner)

- Ø Alguien a quién se le ha transferido la propiedad de una base de datos
- Ø Se le ha otorgado la autorización de crear una base de datos
- Ø Adicione y retira usuarios de la base de datos con SP_ADDUSER
- Ø Otorga y revoca permisos a usuarios para crear objetos en la base de datos y ejecutar comandos con GRANT
- Ø Ejecuta algunas tareas de operador del sistema en su propia Bases de Datos
- Ø Ejecuta CHECKPOINT y DBCC en la Bases de Datos
- Ø Tiene todos los privilegios sobre todos los objetos en la Bases de Datos usando SETUSER
- Ø No pueden otorgar permisos que modifiquen el esquema de un objeto, como CREATE INDEX, ALTER TABLE, DROP OBJECT
- Ø No pueden transferir la propiedad de un objeto

SSO. (Oficial del sistema de Seguridad)

El Oficial del Sistema de Seguridad es responsable de la seguridad del sistema y tiene las siguientes funciones.

- Ø Administración cuentas de usuario.
- Ø Otorgar y revocar los roles (de usuario, sso y oper)
- Ø Otorgan autorización para uso del proxy
- Ø Manejar el sistema de auditoria.
- Ø SSO no puede modificar o borrar logines; esto sólo puede hacerlo un login con rol SA

Oper. (Operador)
- Ø Los Operadores pueden respaldar y cargar todas las bases de datos y logs de transacciones en el servidor
  - Ø Realizan DUMP DB, DUMP TRANSACTION, LOAD DB y LOAD TRANSACTION
  - Ø No tienen que ser propietarios de una base de datos para realizar tareas de mantenimiento
- Ø 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

Introducción a la Seguridad del SQL Server

SQL Server tiene un sistema de seguridad de capas descrito aquí:

Aceso a;		Controlled by:
Server, database	- - à	adding/dropping logins, users
Objects, commands	- - à	granting/revoking permissions

Para controlar el acceso:
- Ø Agregar y/o borrar logines y usuarios de bases de datos
- Ø Agregar y/o borrar grupos y roles de usuarios
- Ø Otorgar y revocar permisos en objetos o comandos a usuarios, grupos o roles

Logines del SQL Server
- Ø Para conectarse a un SQL Server, es necesario un login
- Ø Los detalles de un login son guardados en la tabla syslogins en la base de datos master
- Ø Un rol SSO puede agregar logines usando SP_ADDLOGIN
- Ø La sintaxis de SP_ADDLOGIN es:

```
sp_addlogin login_name, passwd [,defaultdb[, deflanguage [,fullname]]]
sp_addlogin claire, orange, public_db, french, "Claire"
```

- Ø Notas:
  - Ø Los Passwords deben ser por lo menos de 6 bytes; el nulo no esta permitido
  - Ø Especificar una base de datos predeterminada (si no se especifica, master es la predeterminada, por lo que no es buena idea!)

Funciones de Logines
- Ø SA:

```
sp_modifylogin login_name, option, value
sp_droplogin login_name
```

- Ø SSO:

```
sp_addlogin login_name, password,...
sp_password caller_passwd, new_passwd [,login_name]
sp_configure password, n
```

- Ø SA o SSO:

```
sp_locklogin [login_name, "{lock | unlock"}]
```

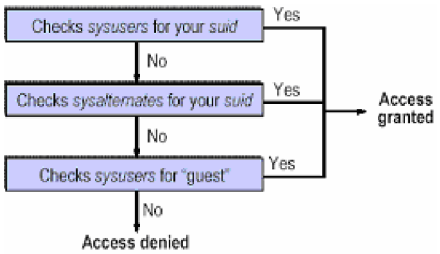
Usuarios de Base de Datos
- Ø Para accesar a una base de datos, el usuario debe estar dado de alta en esa base de datos
- Ø Los usuarios son insertados en la tabla sysusers en cada base de datos
- Ø Para agregar un usuario a la base de datos use la sintaxis:

```
sp_adduser login_name [, name_in_db [, grpname]]
sp_adduser claire, claireb
```

- Ø Solo el propietario de la base de datos puede agregar usuarios
  - Ø Los administradores del sistema automáticamente son dbo en cada base de datos que accesan y por lo tanto pueden agregar usuarios también
- Ø Para desplegar una lista de los usuarios de una base de datos con su nombre de login, use SP_HELPUSER

Acceso a la Base de Datos

Cuando intenta accesar a una base de datos, SQL Server:



Alias

- Puede tratar a más de un login como un mismo usuario de una base de datos, tomando todos sus privilegios
  - Hacer que logines sean dbo u otro nombre de usuario
  - Si las actividades del alias son auditadas, la identidad real lo será también
- Para crear un alias, use SP_ADDALIAS

```
sp_addalias loginame, name_in_db
```

- Agrega un registro en sysalternates en la base de datos
  - Por ejemplo, Mary es dueña de una base de datos; ella quiere que Jane use la base de datos como dueña; Jane tiene un login pero no es usuario de la base de datos de Mary; Mary ejecuta el comando:

```
sp_addalias jane, dbo
```

Revocando Acceso al SQL Server o Base de Datos

- Para borrar cuentas de login, use:

```
sp_droplogin login_name
```

- Para bloquear, desbloquear o desplegar cuentas bloqueadas use:

```
sp_locklogin [login_name, "{lock | unlock}"]
```

- Para borrar un usuario, use:

```
sp_dropuser name_in_db
```

- Para borrar el alias de un usuario, use:

```
sp_dropalias login_name
```

- Para borrar un usuario o alias de una base de datos, debe de estar en esa misma base de datos

Agregando y Asignando Grupos

- Proporcionar una forma conveniente para otorgar y revocar permisos a más de un usuario y nombre colectivo de muchos usuarios. Para agregar un grupo:

```
sp_addgroup grpname
```

- Crea grupos antes de agregar usuarios porque el comando SP_ADDUSER deja que se le asigne un grupo al usuario, por ejemplo:

```
sp_adduser robert, bob, senioreng
```

- Para reasignar o asignar un usuario a un grupo, use SP_CHANGEGROUP:

```
sp_changegroup senioreng, steve
```

- El grupo public esta siempre presente y todos están en él; el usuario sólo puede estar asignado a un grupo

Desplegando Información de Grupos

- Use SP_HELPGROUP para desplegar los grupos en una base de datos use:

```
sp_helpgroup [grpname]
```

  - Sin un parámetro, regresa todos los grupos en la base de datos (incluyendo roles de usuario y de sistema). Con un parámetro, regresa usuarios de ese grupo
- La tabla sysusers tiene un registro para cada grupo
  - Los roles de usuario y de sistema aparecen como grupos en la tabla sysusers de todas las bases de datos aunque trabaja de manera diferente

Funciones Útiles

```
user_id( )
suser_id( )
suser_name( )
db_id( )
db_name( )
user_name( )
```

Roles de Sistema del SQL Server

- Para otorgar esos roles a otros logines, use SP_ROLE. La sintaxis para SP_ROLE es:

```
sp_role "{grant|revoke}", "{sa_role|sso_role|oper_role}", login_name
sp_role "grant", "sa_role", robby
```

- SSOs otorgan y revocan roles SSO y OPER; SAs otorgan y revocan el rol SA
- Al ejecutar SP_ROLE agrega o borra un registro en master..syloginroles
  - Estos toman efecto en el siguiente ingreso al servidor
- Incrementa la responsabilidad de un usuario ya que se auditan sus actividades

- SQL Server no permite bloquear el último login desbloqueado con el rol SA, ni el último login desbloqueado con el rol SSO. SP_DROPLOGIN, SP_LOCKLOGIN y SP_ROLE tienen que checar que se cumpla esta regla

proc_role( )

Ø La función PROC_ROLE checa los roles cuando un procedimiento es ejecutado. Si regresa 1 el usuario posee el rol específico

Ø Un ejemplo usando PROC_ROLE en el procedimiento almacenado test_proc en el que se requiere que lo ejecute un administrador del sistema:

```
if (proc_role("sa_role") = 0)
begin
 print "Usted no tiene el Rol SA."
 return -1
end
else
 print "Usted tiene el ROL SA."
 return 0
```

Activando y Desactivando Roles del Sistema

Ø Al entrar, los roles que han sido asignados a su login son activados automáticamente (si hubiera alguno). Para desactivar un rol, use:

```
set role "{sa_role | sso_role | oper_role}" off
set role "sa_role" off
```

Ø Toma efecto inmediatamente y dura para toda la sesión, a menos que se reactive

Ø Para reactivar un rol, use SET ROLE ... on

Desplegando Información acerca de Logines

Ø SP_DISPLAYLOGIN despliega información acerca de un login, incluyendo que roles le han sido otorgados

```
1> sp_displaylogin sa
2> go
 Suid: 1
 Loginame: sa
 Fullname:
 Configured Authorization: sa_role sso_role oper_role replication_role
 Locked: NO
 Date of Last Password Change: Nov 17 1994 12:02PM
 (return status = 0)\
```

Otorgando Privilegios en Comandos

Ø Use grant para dar permisos en comandos:

```
 Grant {all [privileges] | command_list} to {public | name_list |
role_name}
     grant create rule to fred
     grant create rule, create table to engineering
     grant create database to mary, john  (SA only)
```

Ø Permisos de Comandos sobre bases de datos que no pueden ser dados:

- Ø DBCC, DUMP DATABASE, DUMP TRAN, LOAD DATABASE, LOAD TRANSACTION, SETUSER

Revocando Privilegios en Comandos

Ø Use revoke para quitar permisos de comandos

Ø Sólo un rol SA puede revoke create database

```
 revoke {all [privileges] | command_list} from {public | name_list |
role_name}
     revoke create table from public
```

Otorgando Permisos sobre los Objetos

Ø Use grant para dar permisos en objetos específicos

```
 grant {all [privileges] | permission_list}
 on {table_name [(column_list)]
 | view_name [(column_list)]
 | stored_procedure_name}
 to {public | name_list | role_name}
 [with grant option]
     grant insert, delete on titles to mary, sales
```

Ø Permisos que no pueden ser otorgados: CREATE INDEX, CREATE TRIGGER, ALTER TABLE, DROP TABLE, TRUNCATE TABLE, UPDATE STATISTICS

Revocando Permisos sobre los Objetos

Ø Use revoke para quitar permisos sobre objetos específicos

```
revoke [grant option for]
{all [privileges] | permission_list}
on {table_name [(column_list)]
```

```
  | view_name [(column_list)]
  | stored_procedure_name}
from {public | name_list | role_name}
[cascade]
   revoke insert, delete on titles from mary, sales
```

Opción grant...with grant

Ø Permite algún(os) usuarios otorgar ciertos privilegios a otros usuarios
Ø Se puede dar la opción grant...with grant solo a usuarios individuales, no a public, grupos o roles

```
grant select on mytable to philip with grant option
grant select on mytable to mary with grant option
```

Opción revoke grant...cascade

Quita el permiso select a Philip y a las personas a las que les dio el acceso select.

```
revoke select on mytable from philip, cascade
```

Autorización Create Schema

El comando CREATE SCHEMA crea una colección de objetos que son propiedad de un usuario y ademas puede incluir permisos otorgados en esos objetos. Permite crear y dar permisos de tablas relacionadas en una transacción. Si cualquier sentencia dentro del esquema falla, todo el comando es cancelado (roll back)

Ø Ejemplo:

```
create schema authorization anna
create table tableA (col1 int, col2 int)
create view v1 as select col1 from tableA
grant select on v1 to public
```

Desplegando Permisos

Ø Los comandos que otorgan y agregan privilegios agregan un registro en sysprotects
Ø Para desplegar permisos en un objeto dado, use SP_HELPROTECT

```
sp_helprotect object_name | User
```

Permisos sobre Procedimientos del Sistema

Ø Los administradores del sistema controlan el acceso a los procedimientos y tablas del sistema en master
Ø Características de los procedimientos del sistema
  Ø Residen en sybsystemprocs pero pueden ejecutarse en cualquier base de datos
  Ø Los permisos se fijan en sybsystemprocs..sysprotects por los administradores del sistema
Ø El script installmaster otorga permisos a public para algunos procedimientos del sistema
  Ø Los administradores del sistema pueden revocar esos permisos
Ø Para crear nuevos procedimientos de sistema que todos puedan usar:
  Ø Crearlos en sybsystemprocs y que comiencen con sp_
  Ø Otorgar permisos apropiados y respalde la base de datos

Roles definidos a usuarios:

Ø Las asignaciones de roles en todo el servidor pueden incluir muchos logines– por ejemplo, un rol de contador, maestro.
Ø Son diferentes de los grupos ya que un grupo es específico de una base de datos y se le asocian ids de usuario
Ø Son otorgados a un login específico o a otro rol.
Ø Son administrados dentro de jerarquias de roles para un eficiente manejo de roles
Ø Cualquier login puede tener muchos roles
Ø Una vez definido, los roles pueden ser apagados y prendidos dinámicamente por el usuario, teniendo así mayor flexibilidad con respecto a los grupos

Grupos Versus Roles de Usuarios

Ø Las personas pueden ser parte de un grupo, como de ingeniería, porque trabajan en el departamento de ingeniería
  Ø Un departamento puede contener muchos roles.
Ø Por lo tanto, las personas pueden tener roles de usuario asociados con su login, como:
  Ø senioreng_role, manager_role y teacher_role
Ø El control de acceso puede ser establecido basado en:
  Ø El grupo al cuál pertenece una persona
  Ø El rol que la persona juega en la organización

Limitaciones de los Grupos

Ø Los grupos son internos en una base de datos y no a un servidor, en contraste con los roles de usuario

- Un grupo no puede tener usuarios de muchas bases de datos
- Los permisos pueden asignarse a objetos solo en la base de datos en la cuál el grupo fue creado
- Todos los usuarios son miembros del grupo public; para excluirlos, necesita ejecutar sentencias de revoke explicitas
- Los usuarios pueden ser miembros de solo dos grupos en la misma base de datos, public y otro grupo
- Cuando quita un usuario de un grupo, los privilegios de acceso asociados con sus logines siguen teniendo efecto

Por qué Implementar Roles de Usuario?

- Los roles de usuario cumplen con el estándar ANSI SQL3 para el control de acceso basado en roles y proporcionan extensiones para mejorar su uso como:
  - La jerarquía de roles pueden ser formados para manejar y controlar el acceso
  - Los roles pueden ser definidos para una aplicación específica
  - Los roles pueden ser definidos mutuamente excluyentes, mejorando la seguridad
- Cuatro pasos son requeridos para implementar los roles de usuario:
  - Un login con privilegios de SSO debe crear el rol de usuario.

```
create role role_name [with passwd password]
create role professor_role with passwd publishme
```

  - Después de que el rol es creado, cualquier login con el permiso with grant puede otorgar privilegios de acceso a un rol de usuario.

```
grant update on employee_table to manager_role
grant select on employee_list to manager_role
```

  - El login con privilegios SSO puede otorgar a otros logines la membresía en un rol de usuario.

```
grant role manager_role to mary
```

  - A los usuarios que les fueron otorgados la membresía a un rol deben activar explícitamente sus roles para obtener los privilegios asociados.

```
set role manager_role on
```

Creando Passwords de Roles de Usuario

- Para cambiar un password o un conjunto de passwords después de que el rol de usuario es creado use esta sintaxis:

```
alter role teacher_role add passwd paymemore
```

- El comando falla si un password esta ya asignado
- Para borrar un password (y asignar uno nuevo), use:

```
alter role role_name drop passwd
```

Otorgando Privilegios de Acceso a Roles de Usuario

- Los roles de usuarios son a nivel servidor y se almacenan en la base de datos master y están asignados a los logines
- Se puede otorgar o revocar permisos a los roles de usuarios solo desde la base de datos donde residen actualmente los objetos

Activando el Rol de Usuario

- Asignando un rol de usuario a un login le da la oportunidad al usuario de activar el rol (cuando entra al servidor)
- El login debe activar el rol de usuario para ganar privilegios asociados con el rol
- Para activar un rol de usuario, el login teclea lo siguiente:

```
set role role_name [with passwd password] on
set role ta_role on
set role professor_role with password publishme on
```

Revocando Roles de Usuario de los Logines

- Para revocar un rol de usuario de un login u otro rol de usuario, use esta sintaxis:

```
revoke role role_revoked [, role_revoked, ...]
  from grantee [, grantee, ...]
revoke role ta_role from jharrison
```

- El que otorga puede ser un login u otro rol de usuario
- Debe tener privilegios de SSO para ejecutar esta tarea
- No puede revocar un rol de usuario de un login si a este nunca le fue otorgado el rol.

Borrando un Rol de Usuario

- Un login con privilegios SSO borra el rol

```
drop role role_name [with override]
drop role teacher_role
```

- Si la opción override es usada, la sentencia drop quita los permisos otorgados asociados al rol en cada base de datos o todo el servidor
- Si la opción override no es usada, cualquier permiso otorgado al rol deberá ser revocada directamente antes de ejecutar este comando. De otro manera, se marcará un error
- Cuando se completa con éxito, drop borra todos los miembros logines del rol y borra el rol de cualquier jerarquía de roles en las cuales estaba asociado

Desplegando la Información de un Rol de Usuario

- Un procedimiento del sistema llamado SP_DISPLAYROLES es usado para desplegar:
  - Roles otorgados a un login
  - Roles contenidos por otro rol
  - Roles descendentes para un rol particular
- La sintaxis para sp_displayroles:

```
sp_displayroles [login_name | role_name] [,expand_up | expand_down]
```

Más Procedimientos Almacenados para Roles de Usuario

- Para ver que roles están activos actualmente, use SP_ACTIVEROLES:

```
sp_activeroles [expand_down]
```

Para desplegar información de permisos otorgados a un usuario, el cuál incluye permisos para un grupo o miembro un rol, use SP_HELPROTECT:

```
sp_helprotect name, username, 'grant'
[, 'none' | 'granted' | 'enabled' ] [, rolename]
```

- Esta sentencia regresa una lista de permisos que han sido otorgados a gwashington en student_info con el rol teacher_role:

```
sp_helprotect student_info, gwashington,NULL, 'teacher_role'
```

Funciones Acerca de Roles de Usuario

- Hay varias funciones que obtienen información de los roles:
  - role_contain( ) checa si un rol esta contenido en otro rol; regresa un estado de 1 si el rol es contenido, 0 si no

```
role_contain(role1, role2)
select role_contain ('pay_clerk, 'receptionist')
```

  - role_name( ) regresa el nombre del rol para un srid dado

```
role_name(srid)
```

  - role_id( ) regresa un srid para un nombre de rol dado

```
role_id(role_name)
```

  - proc_role( ) ha sido actualizado para incluir un chequeo en los roles que un login tiene activo
  - show_role( ) despliega cuales roles están activados actualmente para su login

Conducta Predeterminada de los Roles de Usuario

- La conducta predeterminada para todos los roles de usuario es de OFF en el login
- Los roles del sistema están:
- ON en un login si no tiene un password asociado al rol del sistema
- OFF en el login por default si tiene un password asociado al rol del sistema

Qué es la Autorización Proxy?

En versiones anteriores, el comando SETUSER permitía a un usuario tomar la identidad de otro usuario

- Por lo tanto, estaba limitado a una base de datos y solo podía ser usado por SA o dbo
- Muchos clientes tienen entornos en los cuáles muchos usuarios necesitan compartir una cuenta (identidad) a través de muchas bases de datos. Por esta razón, la autorización proxy deja que los usuarios asuman la identidad de otros usuarios en todo el servidor
  - En este caso, los usuarios con autorización proxy pueden ser conocidos como otro usuario en el servidor
  - Los usuarios deben estar en la base de datos para los cuáles están establecidos actualmente y deben tomar la identidad de un usuario que también debe existir en la base de datos

Implementando Autorización Proxy

- El SSO debe otorgar permisos a un usuario para usar la sesión de autorización como sigue:

```
grant set session authorization to <login>
grant set proxy to <login>
```

- En este punto, el usuario puede cambiar la sesión de autorización:

```
set session authorization <login_name>
```

- Cuando los usuarios quieran regresar a sus logines originales, ellos usan el mismo comando como sigue:

```
set session authorization <original_login>
```

Limitaciones de la Autorización Proxy

- El comando SET SESSION AUTHORIZATION/SET PROXY no puede ser usado en una sesión activa
- Un usuario no puede ejecutar SET SESSION AUTHORIZATION en un usuario y desde este punto ejecutar otra autorización proxy en otro usuario
- Para cambiar un nuevo login, el usuario debe retornar primero el login original
- El comando SET SESSION AUTHORIZATION/SET PROXY no puede ser usado en un login que esté bloqueado

Monitoreando y Arreglando Problemas

Monitoreando los Error Logs

El Error Log del SQL Server

- Usualmente localizado en el directorio install en el directorio sybase
- La información es agregada cada vez que se inicia SQL Server y también cuando hay un error fatal o error del kernel
- Ejemplo del formato del Error log:

```
date                    time         sender message
00:00000:00001:1998/04/07 14:44:30.90 server on top default  character set:
00:00000:00001:1998/04/07 14:49:09.75 server DBCC TRACEON 8399, SPID 1
```

Mensajes de Error

- Cuando ocurre un error, SQL Server genera un mensaje de error, el cual contiene: Número de error, Nivel de severidad, estado, nombre del servidor, número de línea que genero el error y un mensaje.
- Los mensajes del sistema proporcionados se almacenan en master..sysmessages: no la altere

Niveles de Severidad

	Nivel de Severidad		
		10	Información del estado, información adicional, no error
Error de usuario (Sintaxis, permisos, etc.)		11	Error de Usuario, sintaxis, objeto no encontrado, violación de permisos, etc.
	No Fatal	--	
		--	
		--	
		16	Error de recursos – informa a SA – puede necesitar reconfiguración
		17	Error interno, falla de software no fatal
Error de SW o HW		18	
Informa SA		19	Proceso único infectado
		20	Proceso de Base de Datos Infectado
		21	Integridad de tabla sospechosa
Problema del Sistema	Fatal	22	Integridad de Base de Datos Sospechosa
		23	Error de HW o corrupción de tabla del sistema
		24	Error interno misceláneo
		25	
		26	

- Los niveles 10 - 16 aparecen solo en la pantalla del usuario
- Monitoree el error log para encontrar errores de software y hardware de nivel de severidad 17 y superiores
  - Los usuarios no pueden reportar errores de severidad 17 y 18 si su trabajo no es interrumpido; necesita monitorear el error log para esos errores
  - Prepare una rutina que navegue por el error log buscando errores números de error específicos o niveles

Tips al Monitorear el Error Log

- El error log crece constantemente y necesitar ser limpiado regularmente
  - Asegúrese de cerrar el servidor primero
  - Haga una copia del error log antes de borrar
- Los usuarios pueden especificar ciertos mensajes que son escritos al error log del SQL Server
- Los procedimientos SP_ADDMESSAGE o SP_ALTERMESSAGE pueden usarse para especificar algún mensaje definido por el usuario que debe ser escrito al error log
- El usuario pueden especificar que los siguientes eventos sean escritos al error log:
  - Exitosas conexiones al SQL Server
  - Fallidas conexiones al SQL Server

Monitoreando el Log del Backup Server

- Es creado cuando es instalado el Backup Server. Por default, en $SYBASE/install/backup.log
- Contiene información del arranque, errores
- Monitorea frecuentemente, después de que cada base de datos haya sido respaldada

Monitoreando el Uso del Espacio Usando Procedimientos Almacenados

Técnicas para el Monitoreo del Uso del Espacio

- Asegure un buen funcionamiento en el entorno de la base de datos, checando que haya espacio suficiente para el log, objetos de base de datos, database objects y todo lo crítico
- Si no hay espacio en el log, las consultas trabajarán, pero las modificaciones no
- Si el espacio en las tablas u otros objetos se termina, entonces si se quiere insertar en estas tablas o crear nuevos objetos no se podrá hacer
- Use las siguientes herramientas y funciones para monitorear el uso del espacio: SP_HELPDB, SP_HELPSEGMENT, SP_SPACEUSED, DBCC CHECKTABLE, MANEJADOR THRESHOLD

sp_spaceused

Despliega el espacio libre dentro de las tablas o en la base de datos

```
1> sp_spaceused titles
2> go
Name   rows reserved data index_size unused
------ ---- -------- ---- ---------- ------
Titles   18    48 KB 6 KB        4 KB  36 KB

1> sp_spaceused
2> go
db_name db_size reserved data index_size unused
------- ------- -------- ---- ---------- ------
  pubs2  2.0 MB  1386 KB  452       94 KB 840 KB
```

Dos Funciones Útiles

- SP_SPACEUSED usa dos funciones que pueden ejecutarse independientemente:
  - DATA_PGS(OBJECT_ID, {DOAMPG | IOAMPG}) estima páginas usadas por tabla
  - ROWCNT(DOAMPG) estima registros en una tabla
- Ejemplo de uso de data_pgs() para syslogs:

```
select data_pgs(8, doampg) from sysindexes where id = 8
```

- SP_SPACEUSED y esas dos funciones son más exactas si se ejecutan después de DBCC CHECKTABLE( )

dbcc checktable

- DBCC CHECKTABLE (tablename) reporta con más exactitud cuántas páginas de datos están en la tabla especificada
- Si el segmento del log está en su propio dispositivo checktable (syslogs) reporta la cantidad de espacio de log usado y qué porcentaje de log está libre

```
Checking syslogs
The total number of data pages in this table is 7.
NOTICE: Space used on the log segment is 0.01 Mbytes, 1.37%.
NOTICE: Space free on the log segment is 0.99 Mbytes, 98.63%.
Table has 174 data rows.
```

dbcc checkcatalog

- DBCC CHECKCATALOG checa la consistencia dentro y entre las tablas del sistema encontradas en una base de datos, verifica que:
  - Cada tipo en syscolumns tenga un registro coincidente en systypes
  - Cada tabla y vista en sysobjects tenga por lo menos una columna en syscolumns
  - El último checkpoint en syslogs sea válido

Procedimientos Thresholds de Última Oportunidad

- Ø Si un log esta sobre su propio segmento, SQL Server automáticamente crea un threshold de última oportunidad que monitorea el uso del espacio en el segmento. Estos previenen desastrosas situaciones al quedarse sin espacio ya que pueden suspender o abortar los procesos del usuarios
- Ø Cuando un threshold de última oportunidad en el log es cruzado (cuando los segmentos de log o datos se han llenado):
  - Ø los procesos del usuario que intenten escribir en el log serán suspendidos o abortados (Por default, son suspendidos)
  - Ø Envía un mensaje de error log por cada transacción suspendida
  - Ø Ejecuta el procedimientos SP_THRESHOLDACTION
- Ø SQL Server no suministra SP_THRESHOLDACTION; lo tiene que escribir
- Ø No olvide otorgar permisos:
  grant exec on sp_thresholdaction to dbo
- Ø Cualquier usuario con el permiso create procedure puede crear un procedimiento threshold
  - Ø Típicamente, el dueño de la base de datos lo crea
- Ø Si el threshold de última oportunidad es cruzado y SQL Server no puede encontrar SP_THRESHOLDACTION, un mensaje de error se va al error log
- Ø Todo el trabajo de esa base de datos se cuelga hasta que el log es respaldado

Ejemplo de un procedimiento threshold que respalda el log de transacciones e imprime un mensaje en el error log:

```
create procedure sp_thresholdaction
@dbname varchar(30),@segmname varchar(30), @space_left int, @status int
as
dump transaction @dbname to "/dev/rmt4"
print "LOG DUMP: '%1!' for '%2!' dumped", @segmame, @dbname
```

- Ø Extiende el log

Thresholds Free-Space

- Ø Crear adicionales thresholds, llamados free-space thresholds, en cualquier segmento de la base de datos (datos o log)
- Ø Varios free-space thresholds pueden estar en un segmento dado
- Ø Crear un procedimiento almacenado asociado con cada free- space threshold
  sp_addthreshold dbname, segname, free_space, proc_name
- Ø para dar de baja un threshold
  sp_dropthreshold
  - Ø Cuando el threshold es cruzado, SQL Server ejecuta el procedimiento almacenado
- Ø De nombres de segmentos que identifique para que son usados y use extensiones como "_seg"

Desplegando Información acerca de los Thresholds

- Ø Use SP_HELPTHRESHOLD para ver información acerca de los thresholds
  sp_helpthreshold [segment_name]
- Ø Sin un nombre de segmento, despliega información acerca de todos los thresholds en la base de datos

```
sp_helpthreshold data_seg
go
segment name free_pages last chance? threshold procedure
------------ ---------- ------------ --------------------
    data_seg        200            0            data_proc
```

Abortando o Suspendiendo Transacciones de Usuario

- Ø Use SP_DBOPTION (en master) para cambiar:
  sp_dboption salesdb, "abort tran on log full", true
- Ø Entonces ejecute CHECKPOINT en la base de datos especificada
- Ø Si su procedimiento threshold vuelca el log de transacciones, ¿es mejor si los procesos del usuario son suspendidos o abortados?
- Ø Preparando un procedimiento threshold espacio-libre que respalde el log minimiza la posibilidad de bloquear a los usuarios y respaldar el log de transacciones cuando sea necesario
- Ø Si el procedimiento escribe al error log usando sentencias PRINT, entonces también puede ser monitoreado

Usando Monitores SQL Server con Sybase Central

Monitoreando SQL Server Usando Sybase Central

Sybase Central es una herramienta administrativa gráfica para los productos Sybase, su estrategia es la de una única consola integrada para servidores y productos. Para que Sybase Central monitoree su entorno, deberá realizar los siguientes pasos:

- Use DSEDIT para configurar el archivo de interfaces para incluir los nombres de los servidores a ser monitoreados
- Agregue el plug-in usando la opción Sybase Central Plug-ins
- Conéctese a SQL Server a través de Sybase Central
- Crear un perfil de conexión

Tareas Administrativas Comunes

Usando Sybase Central, se pueden realizar las siguientes tareas administrativas:

- Conectarse y desconectarse del SQL Server
- Desplegar información acerca del SQL Server y los objetos que controla
- Fijar los parámetros de configuración del SQL Server
- Desplegar procesos de usuario y eliminar procesos
- Manejar servidores remotos
- Se puede usar Sybase Central para:
- Ver el estado de los procesos del usuario en el SQL Server
  - Para desplegar procesos de usuario en el SQL Server, seleccione el folder de procesos (cada proceso tiene un icono)
  - La etiqueta parámetros de la hoja de procesos lista información adicional acerca de los procesos
- Fijar atributos de ejecución de los procesos
  - Cambiar la prioridad de un proceso asignándole a un motor de grupo
  - La etiqueta de atributos de ejecución de la hoja de procesos deja cambiar los atributos a su valor predeterminado
- Eliminar procesos

Monitores del SQL Server

El Monitor del SQL Server consiste en cuatro componentes que generan o despliegan datos de rendimiento:

- Servidor Monitor: Colecciona datos de rendimiento del SQL Server en tiempo real y lo hace disponible a otros componentes monitores del SQL Server
- Servidor Histórico: Obtiene datos de rendimiento del SQL Server del servidor monitor y lo guarda en archivos para su análisis posterior.
- Monitores en el Plug-in SQL Server para Sybase Central (Visor de monitor): Obtiene datos de rendimiento del SQL Server del servidor monitor y despliega los datos en tiempo real en tablas y en gráficos
- Monitor Client Library: Da apoyo a los usuarios quienes están en desarrollo de aplicaciones de monitoreo

Usando Monitores SQL Server con Sybase Central

- Un servidor monitor debe estar configurado en la misma máquina como el SQL Server se quiera monitorear
- El servidor monitor debe ejecutarse
- Un registro de el servidor monitor configurado debe existir en el sql.ini o archivo de servicios de directorio en la máquina donde Sybase Central está corriendo
- La versión correcta de una máquina virtual Java debe estar instalada en su máquina
- En el árbol de Sybase Central, un folder de monitores debe aparecer bajo el icono ASE
- El login del SQL Server debe tener privilegios apropiados

Resultados de los Datos de el Monitoreo

- Para coleccionar el tipo de datos que necesita, se puede fijar lo siguiente:
  - El tipo estadístico: muestra versus datos acumulativos
  - El intervalo de muestra en horas, minutos y segundos
  - Pausando y refrescando datos en la ventana para examinar los resultados
  - Fijar filtros pata limitar los datos a monitorear
  - La siguiente tabla resume los monitores de el SQL Server

Vistazo del desempeño del monitoreó de Sybase Central

Nombre del monitor	Descripción
Cache Monitor	Muestra información acerca del procedure y data cache memoria.

Data Cache Monitor	Nuestra la actividad completa y los niveles de eficiencia de los 10 mayores actividades del data cache (incluyendo name y default).
Device I/O Monitor	Muestra el buffer, actividad I/O en los dispositivos definidos.
Engine Activity Monitor	Muestra detalle de la carga actual del CPU causada por tareas procesándose. El Monitor aísla el tiempo ocupado en el procesamiento versus el tiempo total de CPU usado.
Memory Utilization Monitor	Despliega una grafica de pie mostrando le memoria usada. La información esta basada en los valores de los parámetros configurados que es por lo tanto estático.
Network Activity Monitor	Muestra paquetes de volumen y de tamaño usado para comunicaciones entre el servidor y sus clientes. El monitor también muestra valores de algunos parámetros de configuración que afectan el tráfico de la red.
Object Lock Status Monitor	Muestra detalle de lo actualmente boqueado en las tablas.
Object Page I/O Monitor	Muestra páginas estáticas físicas y lógicas de I/O asociadas con tablas y sus índices. Muestra actividad en todas las tablas incluyendo las del sistema, trabajo temporal y de bases de datos.
Performance Summary Monitor	Resume los principales indicadores de eficiencia incluyendo uso de CPU, índices de transacción, I/O, dispositivos I/O y bloqueos.
Performance Trends Monitor	Grafica estadísticas valores seleccionados por los usuarios. Hasta 60 intervalos de muestra. Cada estadística seleccionada aparece en graficas separadas.
Process Activity Monitor	Muestra información de recursos de alto nivel de procesos actualmente ejecutándose. Si un proceso consiste en hilos de proceso el monitor agrega todos los datos para todos los hilos y presenta solo el total para el proceso padre (el total).
Stored Procedure Activity Monitor	Muestra métricas en ejecución, paginas I/O y bloqueos para todos los store procedures y triggers que han ejecutado durante la muestra o sesión.
Transaction Activity Monitor	Muestra información resumida acerca de las actividades administradas por el servidor.

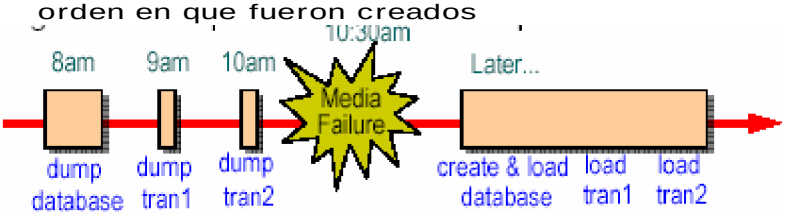
Respaldando Bases de Datos y Logs de Transacciones

¿Qué es un Respaldo?

- Ø DUMP database hace respaldos físicos de toda una base de datos, tanto datos, como el log de transacciones
- Ø DUMP TRANSACTION hace respaldos físicos de el log de transacciones solamente
- Ø No use comandos de copiado del sistema operativo

Cómo restaurar

- Ø Si una falla ocurre, se vuelven a crear las bases de datos cargando los respaldos en el orden en que fueron creados



- Ø Si el log está en un dispositivo diferente, la base de datos puede ser recuperada hasta el tiempo en que fue la falla (10:30 am.). Por otra parte, la base de datos puede ser recuperada hasta el último respaldo de el log de transacciones (10 am)

Respaldos del SQL Server

- Ø Todos los respaldos del SQL Server son realizados por el servidor de respaldos
- Ø Un programa servidor ejecutándose en la misma máquina como un SQL Server
- Ø La arquitectura de respaldo usa el paradigma cliente/servidor, con el SQL Server como cliente de un servidor de respaldo (Backup Server)
- Ø Características
  - Ø Dinámico—Pueden realizarse mientras los usuarios están activos

- Ø Puede respaldar dispositivos locales o dispositivos de otras máquinas
- Ø Puede realizar respaldos multiarchivos (muchas bases de datos en un solo dispositivo)
- Ø Puede realizar respaldos multivolumen (grandes bases de datos en varios dispositivos)
- Ø Puede prepara respaldos automáticos

Backup Server: Respaldos y Cargas Remotas

- Ø Los respaldos remotos son posibles si el Backup Server local se comunica con un Backup Server de una maquina remota
- Ø Un Backup Server local siempre es necesario
  - Ø El Backup Server local puede manejar el dispositivo de respaldo si este dispositivo es conocido por la máquina local
- Ø Si muchos SQL Servers están en muchas máquinas, se tiene que usar un Backup Server remoto sobre una red
- Ø Cada SQL Server tiene que conectarse a un Backup Server local para poder usar un Backup Server remoto
- Ø Si las plataformas local y remotas son homogéneas, los volúmenes escritas local y remotamente son intercambiables

Identificando un Backup Server Local

- Ø SQL Server usa el nombre SYB_BACKUP para referirse al Backup Server local y, por default, se ve este nombre en el archivo de interfaces siempre que se hagan respaldos o cargas
- Ø Si quiere que SQL Server use otro nombre en el Backup Server, especifique el nombre durante la instalación o agréguelo en el archivo de interfaces y use SP_ADDSERVER para hacérselo conocer al SQL Server

```
sp_addserver SYB_BACKUP, null, BK_NUEVONOMBRE
```

- Ø Asegúrese de configurar SQL Server para acceso remoto

Identificando un Backup Server Remoto

- Ø Para identificar un Backup Server remoto, agréguelo en el archivo de interfaces
  - Ø Cuando el nombre de un Backup Server remoto es referido en los comandos dump/load, el Backup Server local lo busca en el archivo de interfaces
  - Ø Recuerde, el registro en la interfase debe ser exacto y formateado correctamente

Backup Server Local y Remoto

- Ø Todas las demandas del Backup Server van al Backup Server local
- Ø Para pasar la demanda del respaldo a un Backup Server remoto, especifique el nombre en el comando DUMP database. Ejemplo:

```
Dump database salesdb To "/dev/rmt4" at B_SHANTI
                  Stripe on "/dev/rmt5" at B_SHANTI
```

En muchas plataformas, use el nombre del camino físico para un Backup Server remoto; los nombres del dispositivo lógico de SYBASE no trabajarán

- Ø Asegúrese que los registros del archivo de interfaces sean exactos para ambos Backup Servers

Manejando Servidores Remotos

- Ø Para tener acceso a un servidor remoto, un SSO puede crear un servidor remoto y entonces definirlo en el SQL Server local
- Ø La instalación del SQL Server puede ser configurada para que un usuario conectado al SQL Server pueda requerir la ejecución de un procedimiento almacenado en otro SQL Server
  - Ø El resultado de esta llamada de un procedimiento remoto (RPC) es regresada al proceso que llamó ejecutándose en el SQL Server en el cuál el usuario está conectado
- Ø Si Component Integration Services (CIS) es habilitado para el SQL server, también se pueden accesar a datos de un servidor remoto como si fuese un servidor local

Configurando Servidores Remotos

- Ø Un manejador de sitio, el cuál es un método predeterminado para manejar la interacción entre servidores locales y remotos, crea una conexión física entre el servidor local y el servidor remoto
  - Ø Entonces se crea una conexión lógica por cada RPC al servidor remoto
- Ø El manejador CIS RPC siempre es usado para conexiones involucrando tablas proxy
  - Ø Se crean conexiones usando funciones Client-Library
  - Ø Su pueden habilitar para usarse con todos los RPCs

Verificando el Setup

- Ø Arranque el Backup Server

```
Unix: startserver -f RUN_B_VIOLET
```

- Configure SQL Server para acceso remoto (si es necesario, reinicie SQL Server)
- Verifique que:
  - El registro del Backup Server local en sysservers tenga un nombre de red exacto
    sp_helpserver
  - El usuario que inició el proceso del Backup Server (usualmente sybase) tenga permisos de escritura para los dispositivos de respaldo
- Tanto el Backup Servers local y remoto debe ser listado correctamente en el archivo interfaces

```
#
 B_VIOLET
      query tcp sun-ether violet 2001
      master tcp sun-ether violet 2001
#
 B_SHANTI
      query tcp sun-ether shanti 9996
      master tcp sun-ether shanti 9996
```

  - El registro Backup Server local es creado cuando SQL Server es instalado
  - El Backup Server remoto se agrega manualmente (si hay)
  - Si los registros son inexactos, SQL Server puede contactar un Backup Server erróneo y escribir o leer de o en dispositivos erróneos

Seleccionando un Dispositivo de Respaldo

- Se prefieren archivos del disco porque
  - Son más rápidos para el respaldo como para la recuperación que las cintas
  - Llegan a ser más baratos para grandes cantidades de datos
  - Desventaja, los respaldos se pierden si el disco falla
- La cintas pueden ser usadas
  - Ventaja, Son más seguras y más portables que los discos
  - Permiten que una biblioteca de base de datos y respaldos del log de transacciones estar fuera de línea

Agregando un Dispositivo de Respaldo (Local Solamente)

- En muchas plataformas, SQL Server instala automáticamente en sysdevices uno o dos alías para los dispositivos de cinta. Ejemplo:

```
tapedump1 /dev/nrmt4
tapedump2 /dev/nrst0
```

- Para agregar más, use SP_ADDUMPDEVICE

```
sp_addumpdevice "tape | disk", "logicalname", "physicalname"
```

  - Puede ser usado en subsecuentes comandos dump o load. Ejemplo:

```
sp_addumpdevice "tape", "tape3", "/dev/nrmt4",300
sp_addumpdevice "disk", "disk1", "/usr/backups/disk.dump"
```

dump database

- Hace una copia respaldo de la base de datos y del log de transacciones
- Dinámico
  - Puede ser hecho mientras los usuarios están activos

```
dump database salesdb to data_dev1
```

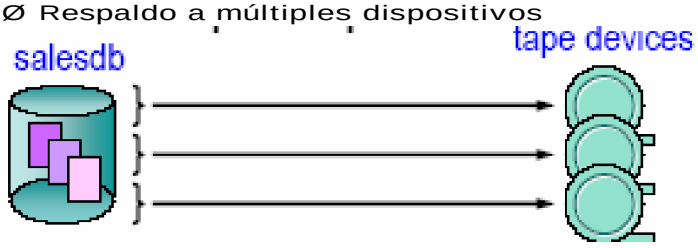
- Que hace dump database
  - Realiza un checkpoint, esto es, los datos y el log son copiados del caché al disco.
  - Copia páginas asignadas (log y datos) para respaldar al dispositivo
  - Captura el estado cercano al final del respaldo. Sintaxis

```
dump database database_name to stripe_device [at b_server_name]
[stripe on stripe_device [at b_server_name]
[with {blocksize = number_bytes, capacity = number_kilobytes,
dumpvolume = volume_name, file = file_name, [nounload | unload],
retaindays = number_days, [noinit | init],
notify = client | operator_console}}]]
```

  - Use las opciones noinit | init y nounload | unload para respaldar varias bases de datos en una cinta

Respaldos Multivolumen o Multidispositivo

- Respaldo a múltiples dispositivos



- Se puede escribir una base de datos a través de múltiples dispositivos concurrentemente usando la opción stripe on

```
dump database db1 to "/dev/nrmt4"
                  stripe on "/dev/nrmt5"
                  stripe on "/dev/nrmt0"
```

- Respaldo a múltiples volúmenes— Cinta solamente
- En el caso de archivos de disco, los volúmenes no pueden cambiarse; sin embargo, SP_VOLCHANGED puede ejecutarse. Los Operadores usan SP_VOLCHANGED para notificar al Backup Server que volumen ha sido montado

Respaldos de Base de Datos Manuales y Automáticos

- Ejecute el comando dump database manualmente regularmente o cuando sea necesario
  - Ventajas
    - Útil en pequeños lugares o cuando es importante saber quién realiza los respaldos
    - Fácil manejo de cintas
    - Útil cuando se necesita saber el nombre del archivo al cargar un volumen multiarchivo
  - Desventajas
    - El personal debe estar disponible cuando se realicen los respaldos
    - El uso de espacio debe estarse monitoreando muy de cerca o el espacio en la base de datos o en el log puede ejecutarse fuera
- Puede hacer que los respaldos se realicen automáticamente usando:
  - Utilidades del Sistema Operativo que pueden ejecutar los scripts de respaldo
  - Manejador Threshold
    - Las utilidades del sistema operativo pueden ejecutar scripts DUMP DATABASE ya sea en cierto tiempo o en intervalos
    - Los sistemas operativos UNIX y NT tienen utilidades de horario CRON que preparan respaldos regulares en cierto horario

dump database: Ejemplos

```
1) dump database pubs2 to "/dev/nrmt0"
2) dump database pubs2 to "/dev/nrmt0" with init
3) dump database pubs2 to "MTA1:", stripe on "MTA2:"
4) dump database pubs2 to "MTA0:" at B_SHANTI
5) dump database pubs2 to dump_dev
```

Asegurando la Consistencia de la Base de Datos

- Aunque no es estrictamente necesario, aquí hay una guía para asegurar la consistencia de la base de datos antes de respaldar la base de datos:
  - DBCC CHECKSTORAGE, DBCC CHECKCATALOG, DBCC CHECKDB, DBCC CHECKALLOC
- Ejecute en el modo single-user; por otra parte, la salida puede indicar las inconsistencias donde no haya ninguna
- No es posible ejecutar estos comandos en modo activo, no se puede ejecutar DBCC CHECKTABLE tablename y DBCC TABLEALLOC para tablas activas
- DBCC CHECKSTORAGE deberá específicamente prepararse en el SQL Server antes de que pueda ser usada

Load database

```
Load database sakesdb from data_dev1
```

- Inicializa páginas no usadas en la base de datos y reemplaza el contenido existente con los datos de la imagen respaldada
- El dueño de la base de datos cargada es el dueño de la base de datos al tiempo de respaldo
- Ejecuta recuperaciones sobre alguna transacción sin commit en el tiempo en que el respaldo se cancelo. Sintaxis:

```
load database database_name from stripe_device [ at b_server_name ]
[, stripe on stripe_device [at b_server_name ]
[with {[dismount | nodismount],[nounload | unload] }]
```

- Ejemplos:

```
load database pubs2 from "data_dev1"
load database pubs2 from "MTA0:"
load database pubs2 from "/dev/rmt4" at B_SHANTI,
                 stripe on "/dev/rmt5" at B_SHANTI
```

Estado Fuera de Línea de la Base de Datos

- Al usar el comando LOAD database automáticamente la base de datos está fuera de línea
  - Elimina la necesidad del administrador de marcar las bases de datos dbo use only cuando se carga una base de datos y el log de transacciones

- Nueva secuencia de carga:
  load database database_name from <pathname_where_dump>
  load transaction database_name from <pathname_where_dump>
- No hay comando que ponga fuera de línea una base de datos manualmente

Comando Online Database
- Ejecutando ONLINE DATABASE hace una base de datos disponible para otros usuarios. Sintaxis:
  online database database_name
- SP_HELPDB database_name despliega si la base de datos tiene un estado online u offline
- ONLINE DATABASE debe ejecutarse desde la base de datos master
- Cuando una base de datos es marcada como online u offline, el servidor emite un mensaje indicando el estado de la base de datos a el error log
- Los dueños de base de datos (dbo's) o usuarios con los roles oper_role o sa_role puede ejecutar este comando

Notas sobre la Carga de una Base de Datos
- Se pueden cargar sólo respaldos hechos en el mismo tipo de máquina
- La base de datos que se va a cargar debe existir y debe ser tan grande como la base de datos respaldada
  - Use SP_HELPDB para ver la cantidad de espacio asignado a una base de datos
- Al cargar datos a una base de datos existente se sobrescriben esos datos; las cargas parciales no son posibles; para crear una nueva base de datos para carga, use la opción for load en CREATE DATABASE
- Si la base de datos es "sospechosa" (estado 256 en sysdatabases), bórrela y vuelva a crearla cargando los datos del respaldo

Dump Transaction
- Hace una copia respaldo del log de transacciones de una base de datos. Ejemplo:
  dump transaction salesdb to saleslog_dev
- Respaldos incrementales, usados después de dump database
  - También el log
- dump database respalda pero no trunca el log de transacciones. Sintaxis:

```
dump transaction database_name to stripe_device [ at b_server_name ]
    [, stripe on stripe_device [at b_server_name ]
    [with { dumpvolume = volume_name, [dismount | nodismount],
    [nounload | unload],  [truncate_only | no_log |no_truncate] } ]
```

- Las mismas opciones de dump database, más opciones de truncado

Que hace el Respaldo de Transacciones
- Respalda todo el log y borra porciones inactivas del log (transacciones con commit)
- Opciones de truncado
  - with truncate_only borra el log sin respaldar
  - with no_log borra el log sin respaldar y no registra la transacción

Que sucede cuando el Log se Llena
- El log de transacciones es añadido cada vez que se modifica la base de datos y puede llegar a llenarse rápidamente
  - Si el log se llena o el threshold (o umbral) del log es cruzado y no trunca el log, todos los trabajos de modificación se detienen
- El log necesita ser truncado para que esto no suceda

Como Truncar el Log
- Existen varias formas de truncar el log:
  dump transaction database_name to stripe_device...
  dump transaction .. with no_log
  dump transaction .. with truncate_only
- Si los respaldos del log de transacción no son guardados, haga que log sea truncado cada vez que ocurre el proceso checkpoint-checking (aproximadamente una vez por minuto)
  sp_dboption dbname,"trunc. log on chkpt.",true
- Si se pone esta opción, respalde frecuentemente la base de datos, porque el log no esta siendo guardado

Transacciones que llenan el Log
  - Actualizando cada registro en una tabla grande
  - Borrando una tabla grande
  - Inserciones basadas en una subconsulta
  - Realizar Bulk copys en tablas grandes que tienen índices

Monitoreando el Log de Transacciones

- Monitoree el log usando las siguientes herramientas:
  - SP_SPACEUSED syslogs
  - select data_pgs (8, doampg) from sysindexes where id = 8
  - DBCC CHECKTABLE (syslogs)
- Use el manejador threshold manager para:
  - Notificarle cuando se llena el log
  - Hace respaldos automáticos después de que cierto threshold ha sido cruzado
- Cheque transacciones que mantienen el log truncándolo cuando es necesario

load transaction

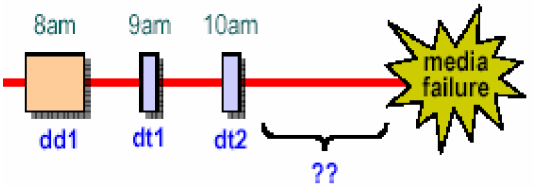
- Lee la anterior copia del respaldo del log de transacciones en el actual log de transacciones para la recuperación. Ejemplo:

```
load transaction salesdb from logdump_dev
```

- Las transacciones deberán ser cargadas en el orden correcto y tienen que estar todas
- DUMP TRANSACTION sin un previo DUMP DATABASE no pasa nada
- LOAD TRANSACTION sin un previo LOAD DATABASE fallará
- Cargados parciales no son posibles

Recuperación Up-to-the-Minute

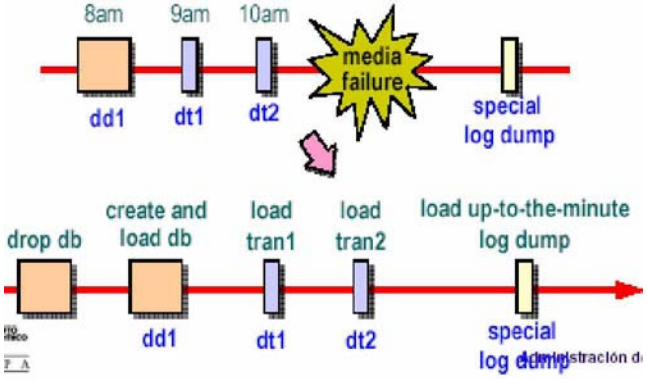
- En el caso de una falla en el medio, puede recuperar cambios que ocurren desde el último respaldo de transacciones?



- Si el log y la base de datos están en el mismo disco físico y ese disco falla, la recuperación de esos cambios no es posible
- Si están en discos separados, tres posibles escenarios de falla en el disco existen
  - Si el disco de la base de datos esta bien, pero el disco del log falla...
  - Si el disco de la base de datos falla, pero el disco del log esta bien...
  - Si fallan el disco de la base de datos y del log...

Recuperando el Log

- Usando dump transaction with no_truncate se obtiene una recuperación up-to-the-minute conocida también como “recovering the log”



Recuperando el Log: Requerimientos

- El Log y la base de datos deben estar en discos separados
- La base de datos master debe estar disponible
- Antes de borrar la base de datos defectuosa, ejecute DUMP TRANSACTION WITH NO_TRUNCATE

Recuperación Point-in-Time

- La recuperación Point-in-time recupera la base de datos en un tiempo o punto particular en el log de transacciones usando una nueva opción de usuario LOAD TRANSACTION
- Puede ser usado por cualquier base de datos en la cuál el log de transacciones puede respaldarse o cargarse
- No se aplica en bases de datos en donde el log y los datos están en el mismo dispositivo
- No puede ser usado en cualquier base de datos que haya tenido un log truncado desde el último DUMP DATABASE

Beneficios
- Recuperación en una sentencia
  - Si una consulta errónea destruye datos, la base de datos puede ser llevada al punto antes que los datos fueran destruidos
- Puede ser usada para servidores con trabajo semipesado

Interfase de Usuario
Sintaxis:
```
load transaction dbname from devname with until_time = "date-time"
```
Ejemplo:
```
1> load transaction employees_db
2> 2> from "/dev/nrmt5"
3> 3> with until_time ="Jan 16, 1997 12:45:59:650PM"
4> 4> go
```
Formato Fecha-Hora
- Mes, día, año hh:mm:ss:ms [am|pm]
- La resolución es más/menos 1/300th de un segundo (siempre que sea especificado en milisegundos)

Usando la Recuperación Point-in-Time
Un usuario accidentalmente borró una tabla importante. (Idealmente, tiene el tiempo exacto en milisegundos antes de que ocurriera el borrado)
1. Use SP_WHO para listar los usuarios que han estado utilizando la base de datos, entonces tiene los usuarios que se han desconectado de la base (Si es necesario, use el comando KILL para eliminar procesos del usuario)
2. Cambie la base de datos al modo single-user
```
1>use master
2>go
3>sp_dboption employees_db "single user", true
4>go
1>use employees_db
2>go
1>checkpoint
2>go
```
3. Respalde el log de transacciones de la base de datos que contiene el error
```
1>dump transaction employees_db to "/dev/nrmt5"
2>go
```
4. Cargue el respaldo más reciente que fue hecho para la base de datos
```
1>use master
2>go
1>load database employees_db from "/dev/nrmt4"
2>go
```
5. Cargue el log de transacciones que fue creado después que la base de datos fuera respaldada por última vez. Entonces cargue el log de transacciones que registra el error usando la opción until_time a un punto antes del error.
```
1>load transaction employees_db from "/dev/ nrmt5"
2>with until_time = "Jan 16 1997 12:45:59:650PM"
3>go
```
6. Permita a los usuarios entrar a la base de datos apagando el modo single-user.
```
1>online database
2>go
1>sp_dboption employees_db "single user", false
2>go
1>use employees_db
2>go
1>checkpoint
2>go
```

La Base de Datos master
La base de datos master contiene información crítica
- Mantenga respaldos recientes de master, especialmente después de:
  - Crear, alterar o borrar bases de datos
  - Agregar o eliminar dispositivos
  - Agregar cuentas y usuarios
- Mantener scripts para realizar esas operaciones
- Si master está dañada o ya no es recuperable, reconstrúyala
  - El disco que mantiene el dispositivo fallido de master
  - SQL Server no puede arrancar
  - DBCC reporta daños
- También debe reconstruir master para duplicarla en otra máquina

Reconstruyendo master con un Respaldo Up-to-Date

- Si tiene respaldos up-to-date:
  - Use BUILDMASTER –M para construir una nueva base de datos master
  - Reinicie el servidor para un solo usuario
  - Agregue un dispositivo de respaldo, si es necesario
  - Cargue master desde un respaldo
  - Reinicie SQL Server con STARTSERVER
  - Cheque la consistencia: ejecutando DBCC CHECKALLOC en cada base de datos y examine las tablas importantes
  - Si todo está bien, reinicie SQL Server para uso multiusuario
- Nunca ejecute BUILDMASTER mientras SQL Server esté funcionando
- Si no tiene respaldos up-to-date, entonces siga las siguientes instrucciones
  - Reagregue dispositivos usando DISK REINIT
  - Reproduzca base de datos creadas usando DISK REFIT
  - Agregar todos los logines y cuentas de usuarios usando scripts isql
  - Reproduzca algunos cambios en model y (tamaño de) tempdb
- Si es más fácil mantenga respaldos up-to-date

Moviendo una Base de Datos a Otro Dispositivo

- Use respaldo y carga para mover una base de datos a un diferente dispositivo

```
disk init name = "device2", ...
dump database salesdb to tape_dev
drop database salesdb
create database salesdb on device2
load database salesdb from tape_dev
```

- Respaldar y cargar a través de plataformas usualmente no es posible
- Vuelva a crear y cargar tablas manualmente usando BULK COPY
- Posibles conflictos:
  - suid, uid
  - Nombre Login
  - Nombre del usuario
- Asegúrese que la base de datos que crea en Server2 coincida con la original
- Asegúrese que el SQL Server destino use el mismo set de ordenamiento y set de caracteres como el original

Buenas Prácticas de Respaldo

- Desarrollar, documentar y probar un conjunto de procedimientos de respaldo y recuperación:
  - Hacer respaldos frecuentes a master
  - Respaldar el log de master frecuentemente
  - Mantener un respaldo reciente de model
  - Hacer respaldos frecuentes de base de datos y log de transacciones para todas las bases de datos
  - Mantenga estadísticas acerca de cuánto tiempo toma un respaldo o una carga y cuanto espacio es requerido
  - Use DBCC para verificar la integridad de sus bases de datos antes de realizar respaldos
  - Respalde la base de datos después de:

```
bcp a alta velocidad
create index
select into
dump transaction with no_log
dump transaction with truncate_only
```

  - Automatizar donde sea apropiado usando procedimientos threshold y scripts que ejecuten respaldos
  - Asegúrese que su archivo de interfaces file esté correcto
  - Catalogue y etiquete su medios de respaldos cuidadosamente

Otras Formas de Protegerse contra Fallos de Medios

- Dispositivos duplicados
- Use la replicación del servidor

Mejorando la Eficiencia Ajustando la Memoria

Cachés y Grandes I/Os

- Cuando se reinicia un SQL Server por primera vez, sólo existe un caché, el default data cache

- Si quiere usar cachés, créelos, especificando: La cantidad de memoria, un nombre. un estado (mixed, log only)
- Se pueden vincular objetos a los cachés o al default data cache
- Se pueden crear, modificar y eliminar bancos de buffer para esos cachés
- Los bancos de Buffer son listas ligadas de páginas de 2K que pueden tener I/O de 2K, 4K, 8K o 16K
- Proporciona un mecanismo para realizar grandes I/Os

Descripción
- Reducen la contención de los recursos del manejador de buffer en un entorno SMP
- Permite un mejor uso de memoria al particionar la memoria a través de líneas
- Tamaño del Banco del Buffer (grandes I/Os)
- Reduce el costo de I/O leyendo y escribiendo múltiples páginas de una base de datos en un sencillo I/O

Default Data Cache
- Siempre hay un default data cache
  - No necesita agregar cachés
- El default data cache es usado:
  - Cuando no se han especificado cachés para objetos
  - Para la recuperación, LOAD DATABASE y LOAD TRANSACTION
- El default data cache es creado cuando SQL Server reinicia y es un caché mixto
- No puede eliminar el default data cache, pero si se permite cambiar su tamaño
- El default data cache siempre tiene un banco del buffer donde se realizan I/Os de 2K; el tamaño mínimo de este banco es de 512K

Use la siguiente instrucción para consultar caches

```
1> sp_helpcache
2> go
Cache Name          Config Size Run Size Overhead
------------------ ----------- -------- --------
accounts                2.00 Mb  2.00 Mb  0.12 Mb
cache1                  8.00 Mb  8.00 Mb  0.41 Mb
cache2                  4.00 Mb  4.00 Mb  0.22 Mb
default data cache      0.00 Mb 37.30 Mb  1.87 Mb

Memory Available For  Memory Configured
Named Caches          To Named Caches
--------------------  -----------------
            52.04 Mb             14.00 Mb

There is 38.04 Mb of memory left over that will be allocated to the default
cache

----------- Cache Binding Information: --------------------

Cache Name Entity Name Type Index Name
---------- --------------- ----------
accounts   pubs2.dbo.sales  table
cache2     pubs2.dbo.titles table
(return status = 0)
```

Ø sp_helpcache

```
1> sp_helpcache "20M"
2> go
  1.00Mb of overhead memory will be needed to manage a cache of size 20M
  (return status = 0)
```

Ø sp_help

```
1> sp_help pubs2..authors
2> go
buffer manager cache binding 1 cache_test1 NULL
```

Use la siguiente instrucción para crear caches

```
1> sp_cacheconfig pub_cache, "10M"
```

Usando el archivo de configuración

```
[Named Cache: pub_cache]
cache size = 10M
cache status = mixed cache
```

Use la siguiente instrucción para modificar caches, para cambiar el tipo o el tamaño

```
sp_cacheconfig pub_cache, logonly
```

Usando el archivo de configuración

```
[Named Cache: pub_cache]
cache status = log only cache
```

Vinculando y Desvinculando Objetos a Cachés

- Puede vincular bases de datos, tablas, índices y logs a cachés
- Un índice (clustered y nonclustered) puede vincularse a un caché diferente de los datos de ese índice
- Una entidad puede vincularse sólo a un caché
- Un caché puede tener varios objetos vinculados a él
- Debe estar en master para vincular y desvincular bases de datos

```
sp_bindcache "pub_cache","pubs2","authors"
sp_bindcache "pub_cache","pubs2", "titles","titleind"
sp_unbindcache "pubs2","authors"
```

- Las I/Os posteriores serán realizadas en el caché predeterminado
- Las acción toma efecto inmediatamente sin requerir la reiniciación

Eliminando un Caché

- Al eliminar un caché se tiene un caché de memoria disponible
- Desvincule objetos antes de eliminar un caché
  - Usando isql
    - sp_cacheconfig "pub_cache", "0"
  - Usando el archivo de configuración
    - Borre el registro del banco del buffer en el archivo de configuración

```
Named Cache:
pub_cache
Cache Status = log only
cache
```

Implicaciones de la Configuración del Caché para la Recuperación

- Sólo el default data cache esta disponible para la recuperación
- El más pequeño default data cache, tomará más tiempo en recuperar
- Consideraciones en el tamaño del Default data cache
  - Mezcla de transacciones
  - Intervalo de recuperación
  - Tamaño total de todos los cachés configurados
- Para aplicaciones con un alto grado de transacciones, asegúrese que el tamaño de el default data cache sea lo suficientemente grande para que la recuperación se haga dentro de un tiempo razonable

Configurando Cachés Metadatos

- Se puede configurar el tamaño de una nueva vía de metadatos con los siguientes parámetros en el SP_CONFIGURE:
  - number of open indexes fija el número máximo de índices abiertos que pueden estar activos en un momento
  - number of open objects fija el número máximo de objetos abiertos que pueden estar activos en un momento
  - number of open databases fija el número máximo de bases de datos abiertos que pueden estar activos en un momento
- Asegúrese de dejar suficiente espacio para el crecimiento y una alta actividad no anticipada
- Estos parámetros de configuración no son estáticos; por consiguiente, deberá restaurar el servidor antes de que los cambios tomen efecto

Planeando la Configuración del Caché Metadato

sp_countmetadata. Se usa para buscar el número total de objetos que corresponden a cada tipo de metadatos. Despliega el número de objetos del tipo especificado así como la cantidad de memoria que requiere el caché metadato

sp_monitorconfig. Usado para reportar el número de descriptores del objeto actualmente en uso y libre, así como el número máximo usado desde la última reinicialización del servidor

sp_helpconfig. Despliega cuánta memoria es requerida para un número de objetos dado

Grandes I/Os

Grandes I/Os minimizan las I/O físicas reduciendo el número de veces que el SQL Server obtenga del disco páginas de datos

Grandes I/Os dan mejores resultados tomando ocho páginas de 2K en una sola operación, en lugar de tomar páginas de 2K, que hace la operación ocho veces separadas

¿Porqué tener Muchos Tamaños de Bancos de Buffer?

- Múltiples tamaños de bancos de buffer son usados para guardar grandes I/Os

- Típicamente, grandes I/Os de datos son más útiles en aplicaciones DSS como:
  - Realizar tareas de mantenimiento regulares como BCP y DBCC
  - Actualizar un millón de registros en un proceso
  - Unir tablas grandes
  - Cargar datos en una tabla usando sentencias insert
  - Buscando en una tabla de 100,000 - registros
  - Realizar un rango de búsqueda en una tabla
- Grandes I/Os son útiles cuando el disco del log es un cuello de botella, entonces, se cambia el tamaño de las I/Os del log

Bancos de Buffer de 2K I/O Obligatorios
- Cada caché contiene un banco del buffer de 2K I/O
- El banco de 2K I/O es usado para utilidades internas

Creando un banco del buffer
Crear el banco del buffer en un caché o en el default data cache
- Ejemplo: Crear un banco del buffer de 4K I/O, con un total de 4 megabytes dentro de un caché

```
sp_poolconfig pub_cache, "4M", "4K"
```

Modificando un Banco de Buffer

```
sp_poolconfig pub_cache, "5M","4K","16K"
```

Disminuye el tamaño banco de 16K I/O en un 1M y se agrega 1M al banco de 4K I/O

Por qué Modificar un Banco del Buffer?
- El banco del buffer es demasiado pequeño y las consultas deben esperar o usar diferentes tamaños de bancos
- El banco del buffer es demasiado pequeño; y puede haber más I/Os físicas debido a que las páginas están siendo limpiadas del caché
- El banco del buffer es demasiado grande y la memoria puede ser utilizada mejor en otra parte
- Se tiene una aplicación que requiere el proceso OLTP durante el día y proceso DSS durante la noche
  - Esto requiere diferentes configuraciones de bancos de buffer— 16K I/O para DSS y 2K I/O para OLTP

Borrando un Banco del Buffer
Borre un banco del buffer que fue usado sólo para dar una solución temporal en un proceso por lote de 16K I/O y que no será necesario en un futuro

```
sp_poolconfig pub_cache, "0", "16K"
```

Usando el archivo de configuración
Elimine el registro del banco del buffer en el archivo de configuración

```
[16k I/O buffer pool]
  pool size = 7M
  wash size = default
```

Restricciones para Crear, Modificar y Borrar Bancos de Buffer
- La suma de todos los buffers en el banco debe ser menor que el tamaño del caché en la memoria total
- No se puede borrar el banco del buffer de 2K en ningún caché
- La I/O para un banco del buffer puede ser especificado en incrementos de 2K, 4K, 8K o 16K
- Un nuevo banco es creado usando memoria del banco predeterminado de 2K I/O pool, a menos que se especifique otro
- El tamaño mínimo de un banco es de 512K
- El banco del buffer máximo para cualquier caché es de 16K

Configurando Cachés desde el Archivo de Configuración
- Estos procedimientos del sistema configuran cachés y grandes I/Os:
  - SP_BINDCACHE, SP_UNBINDCACHE, SP_UNBINDCACHE_ALL, SP_POOLCONFIG, SP_CACHECONFIG, SP_HELPCACHE
  - Estos procedimientos almacenados escriben los cambios en el conjunto de parámetros en el archivo de configuración
- El archivo de configuración editable:
  - Mantiene memoria de cachés y bancos de buffer para el manejador del buffer
  - Reside en el directorio $SYBASE
- Ejemplo de un archivo de configuración:
  - Muestra que hay dos cachés: accounts y cache1

```
[Named Cache:accounts]
 cache size = 2M
 cache status = mixed cache
[Named Cache:cache1]
 cache size = 8M
 cache status = mixed cache
[2K I/O Buffer Pool]
 pool size = 2048K
 wash size = 1534 K
[4K I/O Buffer Pool]
 pool size = 512K
 wash size = 100 K
```

Verificando que Banco del Buffer esta siendo Usado

Ø Use SET SHOWPLAN y SELECT... PREFETCH

```
1> set showplan on
2> go
1> select * from cachtb42
2> (index cacha_ind prefetch 8)
3> go
 QUERY PLAN FOR STATEMENT 1 (at line 1).
 STEP 1
  The type of query is SELECT.
  FROM TABLE cachtb42
  Nested iteration.
  Index : cacha_ind
  Ascending scan.
  Positioning by key.
  Keys are: a
  Using I/O Size 8 Kbytes.
  With LRU Buffer Replacement Strategy.
```

Use el comando set statistics io

```
1> set statistics io on
2> go
1> select count(a) from cachtb42
2> (index cacha_ind prefetch 8)
3> go
------
 100

Table: cachtb42 scan count 1, logical reads: 1, physical reads: 0
Total writes for this command: 0
(100 rows affected)
```

Resumen

Ø Múltiples cachés eliminan la contención spinlock en un entorno SMP
Ø Múltiples bancos del buffer ayudan a controlar recursos de memoria en aplicaciones OLTP, DSS y mixtas
Ø Tomando las páginas frecuentemente más usadas en memoria incrementa el rendimiento, porque el acceso a memoria es más rápido que el acceso al disco
Ø Los cachés Metadato eliminan la necesidad de los cachés en sysindexes, sysdatabases y sysobjects
Ø Grandes I/Os reducen costos de I/O leyendo y escribiendo muchas páginas de la base de datos en una sencilla I/O
Ø Use SET SHOWPLAN, SET STATISTICS IO para verificar que banco del buffer esta siendo utilizado

Introducción al Acceso Paralelo

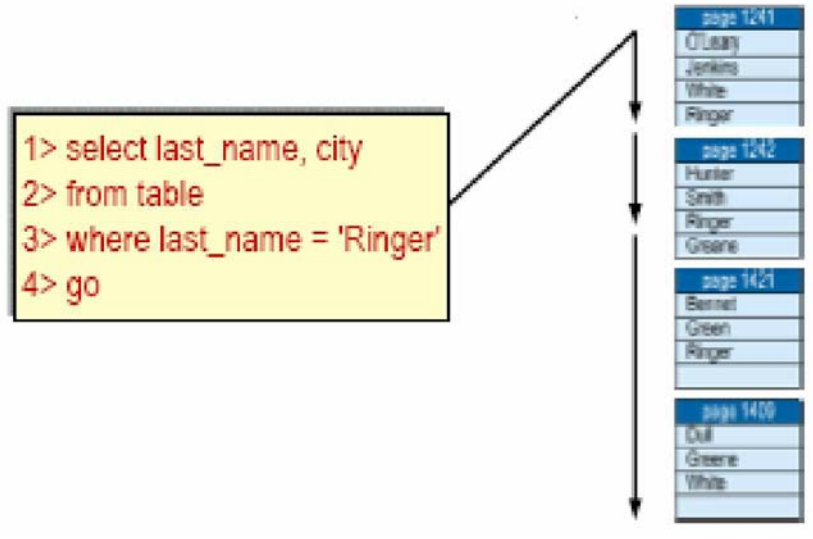
Métodos de Acceso

Los métodos de Acceso son mecanismos de búsqueda de datos en tablas de base de datos relacionales para satisfacer consultas

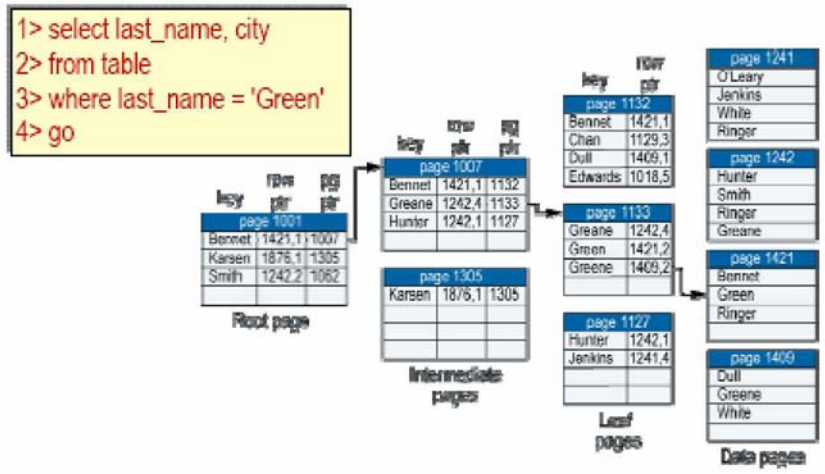
Ø Para muchas consultas, diferentes métodos de acceso son posibles, Por ejemplo, los datos que se necesitan por una consulta a una tabla pueden ser leídos:
  Ø Buscando en toda la tabla
  Ø Buscando en índices clustered
  Ø Buscando en uno o más índices nonclustered
Ø El optimizador de consulta es responsable de estimar los costos de cada método de acceso con relación a otros y al final escoge el método óptimo

Accesando a Datos sin un Índice

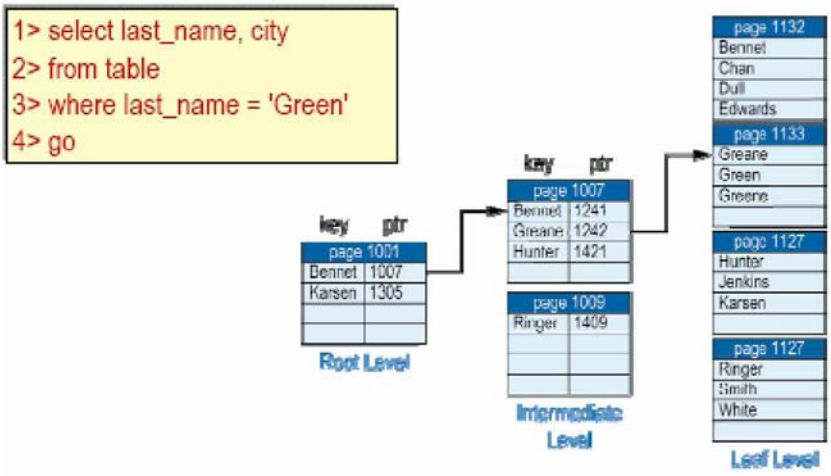
Ø SQL Server almacena tablas sin un índice clustered como cadenas de páginas directas
Ø La tabla, la cuál consiste en páginas ligadas, es llamada comúnmente un tabla heap



Accesando a Datos con un Índice Nonclustered



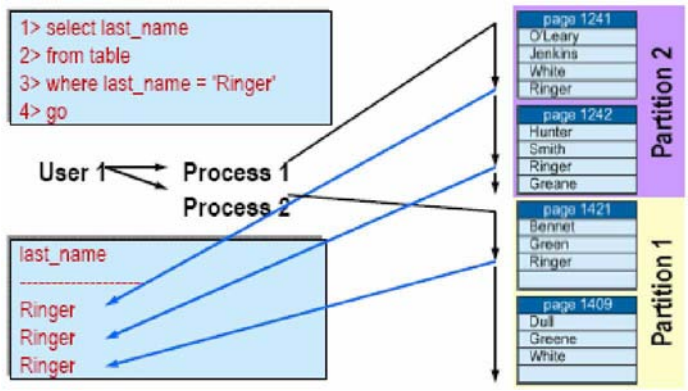
Accesando a Datos con un Índice Clustered



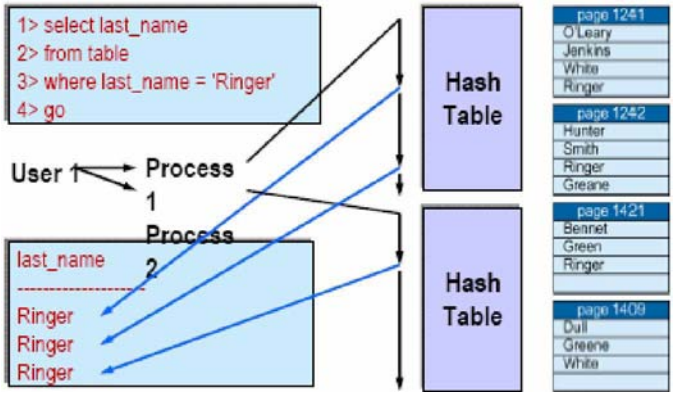
Acceso Paralelo

Ø SQL Server deja que cada usuario use el acceso paralelo o más de un proceso, para una consulta sencilla:
  Ø Búsqueda en particiones
  Ø Búsqueda en Hash

Búsqueda en Particiones



Búsquedas Basadas en Hash



Un Vistazo a la Búsqueda en Paralelo

Fase de Acceso

1. Ejecuta procesos en paralelo
2. Crea un conjunto de resultados o encapsulado, para cada proceso

Fase de Unión

3. Une las cápsulas hacia un solo conjunto de resultados
4. Envía los el conjunto de resultados al cliente

Búsqueda Paralela Worker Processes

- Ø Todos los procesos en el diagrama son worker process
- Ø Cada worker process tiene un ID de proceso único (spid)

Combinación de Búsqueda Paralela

- Ø Un merge es el proceso de combinar múltiples conjuntos de resultados de una consulta paralela hacia un sólo conjunto de resultados
- Ø Hay dos tipos de combinación paralela
  - Ø One-time-only merge
  - Ø Continuous merge
- Ø Algunas consultas paralelas pueden regresar registros en un orden diferente de una ejecución a otra.

Activando el Acceso Paralelo

- Ø Para que el optimizador considere algún método de acceso paralelo:
  - Ø El procesamiento paralelo debe estar activado en el servidor vía SP_CONFIGURE y los parámetros:
    - Ø El parámetro max parallel degree debe ser 2 o mayor
    - Ø El parámetro parallel_degree debe ser 2 o mayor
    - Ø El parámetro number of worker processes debe ser 2 o mayor
    - Ø select into/bulk copy/pllsort debe ser verdadera

Identificando Worker Processes con sp_who

```
1>sp_who
2> go

fid  spid status    cmd
---- ------------- ----------------
NULL  1 runnable...SELECT
```

```
NULL   2 sleeping...NETWORK HANDLER
NULL   3 sleeping...DEADLOCK TUNE
NULL   4 sleeping.. MIRROR HANDLER
NULL   5 runnable...HOUSEKEEPER
NULL   6 sleeping...CHECKPOINT SLEEP
NULL   7 running..  SELECT
8      8 lock sleep.SELECT
8      9 lock sleep.WORKER PROCESS
8     10 send sleep.WORKER PROCESS

(13 rows affected, return status = 0)
```

Búsqueda Paralela y Rendimiento

Las búsquedas paralelas permiten a varios worker processes ejecutar la consulta simultáneamente, lo cuál incrementa tanto la utilización del CPU como las I/Os

- Los factores que mejoran el tiempo de ejecución paralela incluyen:
- Múltiples CPUs
- Múltiples dispositivos de I/O
- La siguiente tabla muestra la escala del CPU en la búsqueda de una tabla de 800MB con 30 particiones, usando 16K I/O

```
  Elapsed
Eng    Time  CPU           I/O Saturation  Throughput
      (secs) Utilization      per Device
---   -----  -----------   --------------  ----------
  1     207       100  %     Not saturated .13 MB/sec
  2     100        98.7%     Not saturated .27 MB/sec
  4      50        98.0%     Not saturated .53 MB/sec
  8      27        93.0%    100% saturated .99 MB/sec
```

Acceso Paralelo y Sentencias SQL

El proceso de consulta paralelo mejora potencialmente el rendimiento de los siguientes tipos de consulta:

- Sentencias select que buscan muchas páginas pero regresan pocos renglones, como buscar en tablas o en índices clustered
- Sentencias select que incluyan group by
- Sentencias select que incluyan funciones agregadas
- Sentencias select que incluyan union, order by o distinct
- Consultas select into
- Sentencias create index

Determinando el Método de Acceso usando showplan

```
  1> select * from table1
  2> go
  QUERY PLAN FOR STATEMENT 1 (at line 1).
  Executed in parallel by coordinating process and 4 worker processes.
   STEP 1
   The type of query is SELECT.
   Executed in parallel by coordinating process and 4 worker processes.
   FROM TABLE
    table1
   Nested iteration. Table Scan. Ascending scan.
   Positioning at start of table.
   Executed in parallel with a 4-way partition scan.
   Using I/O Size 2 Kbytes.
   With LRU Buffer Replacement Strategy.

Parallel network buffer merge.
```

Mejorando la Eficiencia de las Bases de Datos por Medio de Auditorias

Verificando Consistencia de Datos

- La integridad de los datos no sólo puede ser comprometida por el DBMS también por el sistema operativo y el hardware
- El performance del CHECKDB y CHECKALLOC varía grandemente y generalmente está alrededor de una a cinco horas por gigabyte (GB) de datos
- Cuando esta proporción puede ser bastante para las bases de datos de 1 a 2GB, el costo para las bases de datos más grandes a menudo:
  - Impide estos chequeos en todo el sistema
  - Fuerza a medidas extremas para trabajar alrededor de este problema

- Otro problema con estos comandos es la información falsa o rara, errores debido a la actividad de actualización
  - Esto deja a menudo a los usuarios inseguros sobre el nivel de integridad de los datos después de ejecutar los dbcc checks

Técnicas de Validación de la Integridad de la Base de Datos

- SQL Server asegura que las estructuras internas de cada base de datos sean consistentes, pero pueden llegar a ser inconsistentes
- Use DBCC (Database Consistency Checker) como tarea regular para monitorear la integridad de las bases de datos y objetos
- Muchos sitios VLDB no están habilitados para realizar chequeos de consistencia, es mejor usar dbcc paralelos

dbcc checkstorage: Apreciación global

- El comando DBCC CHECKSTORAGE identifica los problemas de:
  - Bajo performance en el chequeo de Consistencia
  - Informa Errores esporádicos
- El comando CHECKSTORAGE se proporciona como una adición además de los DBCC.
- El comando CHECKSTORAGE difiere de los chequeos anteriores:
  - Requiere una base de datos especial
  - Ejecuta en paralelo usando worker processes
  - Mínimiza bloqueos en tablas durante el proceso
  - La integridad se reporta en una base de datos en lugar de un archivo de salida
  - No usa números de error
  - Los fallos en la integridad de los Logs en una base de datos son enviados a la salida estándar
  - Requiere de un manejo dedicado de la base de datos a ejecutar
  - Proporciona procedimientos dbcc para la administración y análisis
  - No reporta errores espurios
  - Realiza algunos chequeos que no hacen CHECKALLOC o CHECKDB
  - No verifica la consistencia de los índices
- Antes de que usted pueda usar los comandos DBCC CHECKSTORAGE, usted debe poner apropiadamente al Adaptive Server de acuerdo a los siguientes siete pasos:
  1. Planear recursos con SP _PLAN_DBCCDB
  2. Instalar la base de datos dbccdb
  3. Configurar cachés para dbccdb
  4. Agregar segmentos de trabajo
  5. Crear espacio de trabajo para dbccdb
  6. Configurar las bases de datos destino con SP_DBCC_UPDATECONFIG
  7. Evaluar la instalación con SP_DBCC_EVALUATEDB
- Los procedimientos SP_ADDSEGMENT, SP_CONFIGURE, SP_CACHECONFIG, SP_POOLCONFIG, SP_DBCC_CREATEWS y SP_DBCC_ALTERWS y los comandos DISK INIT, CREATE DATABASE
  - La ejecución del script $SYBASE/SCRIPTS/INSTALLDBCCDB
- El resultado de esto es:
  - Una base de datos dbccdb instalada en los dispositivos apropiados
  - El tamaño correcto scans y text workspaces
  - El tamaño correcto de cache

Uso de dbcc checkstorage

Ejecutando los chequeos

- La utilidad de checkstorage puede invocarse en cualquiera de estas maneras:
  - Vía el comando DBCC CHECKSTORAGE
  - Vía el procedimiento SP_DBCC_RUNCHECK DBCC

Analizando Resultados

- Una vez que todo está completo y el DBCC CHECKSTORAGE se corre, pueden analizarse los resultados del chequeo vía estos procedimientos:
  - SP_DBCC_FAULTREPORT, SP_DBCC_FULLREPORT,
  - SP_DBCC_STATISTICSREPORT, SP_DBCC_SUMMARYREPORT,
  - SP_DBCC_DIFFERENTIALREPORT

Mantenimiento de la base de datos dbccdb

- Use estos procedimientos dbcc para el mantenimiento requerido:
  - SP_DBCC_DELETEDB, SP_DBCC_DELETEHISTORY, SP_DBCC_CONFIGREPORT, SP_DBCC_UPDATECONFIG
- Usted puede usar otros comandos para dar mantenimiento a dbccdb. Éstos incluyen:

- Ø ALTER DATABASE, DUMP TRANSACTION, DUMP DATABASE, LOAD TRANSACTION y LOAD DATABASE

Antes de que el dbcc checkstorage pueda Correrse

- Ø El comando DBCC CHECKSTORAGE requiere:
  - Ø Recursos especializados
  - Ø Una serie de pasos completados
- Ø Para usar el comando DBCC CHECKSTORAGE involucra completar las dos fases siguientes:
  - Ø Fase 1: Planeación de requerimientos de recursos
  - Ø Fase 2: Crear e inicializar la base de datos dbccdb

Fase 1: Planeando Requerimientos de Recursos

- Ø El comando checkstorage requiere varios tipos de recursos que necesitan asignarse correctamente en términos de tamaño y localización. Los recursos requeridos dependen de:
  - Ø Tamaño de la base(s) de datos que serán checadas
  - Ø Número de bases de datos checadas concurrentemente
  - Ø Tiempo en el cuál el chequeo debe completarse
- Ø La asignación inapropiada de recursos puede causar que el checkstorage falle con un error o llevar a un performance pobre
- Ø Entre los recursos que necesitan ser configurado apropiadamente están:
  - Ø Dispositivos Lógicos
  - Ø Bases de Datos
  - Ø Espacio de Trabajo
  - Ø Cachés
  - Ø Procesos Worker

Usando el procedimiento sp_plan_dbccdb

- Ø SP_PLAN_DBCCDB sirve para ayudar a el usuario recomendando los valores para:
  - Ø Qué dispositivos son convenientes para la base de datos dbccdb
  - Ø El tamaño de base de datos dbccdb, incluyendo tamaño para datos y log
  - Ø Tamaño para el scan y text workspace
  - Ø Tamaño de cache
  - Ø El número máximo de worker processes por checkstorage
- Ø El procedimiento SP_PLAN_DBCCDB basa sus recomendaciones en información de las tablas sysdatabases, sysusages y sysdevices

```
1> sp_plan_dbccdb db1
2> go
Recommended size for dbccdb database is 9MB (data = 7MB, log = 2MB)
  Recommended devices for dbccdb are:

Logical Device Name  Device Size (KB)
-------------------  ----------------
dbdev1                          4000
dbdev2                          8000

Recommended values for workspace size, cache size and process count are:

Dbname scan ws text ws cache process count
------ ------- ------- ----- -------------
db1         96K     48K 1280K             2
```

- Ø SP_DBCC_PLANDB toma el nombre de la base de datos destino como un argumento y regresa los valores recomendados para:
  - Ø Espacio de trabajo, número de procesos worker y tamaño del caché
- Ø El SP_PLAN_DBCCDB debe correrse contra cada base de datos para la que usted planea ejecutar DBCC_CHECKSTORAGE

Planeando recursos: Physical Devices

- Ø Los dispositivos físicos seleccionados para la base de datos dbccdb son críticos con respecto a la actuación del checkstorage
- Ø Idealmente, Los dispositivos físicos escogidos no debe compartirse con cualquier otra base de datos en el Adaptive Server
- Ø Si todos los dispositivos físicos existentes están en uso, el procedimiento SP_PLAN_DBCCDB especifica que ningún dispositivo existente es conveniente
- Ø Cualquier dispositivo puede usarse, pero la falla de no usar un dispositivo físico especializado puede producir una disminución del performance
- Ø Si usted no puede dedicar un dispositivo físico al dbccdb, cree la base de datos en el dispositivo(s) con la menor actividad de I/O

- El dispositivo master no debe usarse para la base de datos del dbccdb

Planeando recursos : Database Size

- El comando DBCC CHECKSTORAGE requiere de una base de datos llamada dbccdb
- Para un mejor rendimiento, esta base de datos debe estar sobre uno o más dispositivos
- El tamaño de la base de datos dbccdb puede ser determinado por el procedimiento SP_PLAN_DBCCDB y el comando CREATE DATABASE es usado para crearla
- Una vez creada, el script $SYBASE/SCRIPTS/installdbccdb debe ejecutarse para:
  - Crear e inicializar las tablas del sistema dbcc
  - Crear los procedimientos almacenados dbcc
- La base de datos dbccdb necesita ser bastante grande para guardar:
  - Scan & text workspaces para ejecuciones concurrentes de checkstorage
  - Errores y datos de estadística informados por checkstorage
- El procedimiento SP_PLAN_DBCCDB calcula el tamaño de la base de datos recomendado que usan las siguientes funciones:
  - No más de dos checkstorage funcionando al mismo tiempo
  - Guarda los Workspaces para las dos bases de datos más grandes
  - La base de datos dbccdb guarda datos de un promedio de cinco ejecuciones
  - Los datos Fault promedian alrededor de 50K por la ejecución
  - El datos de estadística promedia alrededor de 500K por la ejecución
- El tamaño mínimo para la base de datos del dbccdb es 4MB

Ajustando el tamaño de la base de datos

- El tamaño de base de datos de dbccdb reales pueden ser más altos si:
  - Más de cinco ejecuciones se guardan en dbccdb
  - Más de dos concurrentes checkstorage se ejecutan
- Para ambientes que están cortos en espacio de disco, usted puede usar un tamaño de base de datos dbccdb más pequeño que el tamaño recomendado por SP_PLAN_DBCCDB si:
  - Sólo una ejecución de checkstorage se ejecuta a un momento
  - Menos de cinco ejecuciones y valores de Fault y datos de estadísticas se salvan
- Cuando es creada dbccdb, dos espacios de trabajo deben ser creados:
  - Un Scan workspace contiene un registro por cada página de la base de datos destino
  - El texto workspace contiene un registro por cada tabla en la base de datos destino que contenga columnas de texto o de imagen
- Cada workspace debe tener un nombre único. Sintaxis:

```
sp_dbcc_createws dbname, segname, [ wsname ], wstype, "wssize[K|M]"
```

Planeando recursos: Transaction Log Size

- El procedimiento SP_PLAN_DBCCDB siempre recomienda el tamaño del log de transacciones a 2 MB
- Mientras este valor es suficiente en la mayoría de las situaciones, como todos los logs de transacción, puede llenarse
- Las únicas actividades que los logs del checkstorage son las inserciones de estadísticas y los datos Fault- relacionados
- Si un número grande de Faults se descubre encima de un corto intervalo de tiempo, el log de transacción puede llenarse
- Si el log se llena, el checkstorage se suspende hasta que el espacio del log sea suficiente
- Si el trabajo suspendido es terminado por el usuario, antes de que escribiera a la base de datos la dbccdb será accesible

Scan workspace

- Los scan workspace requiere 18 bytes de almacenamiento para cada página en la base de datos designado (aproximadamente 1% del tamaño de la base de datos)
- El procedimiento SP_PLAN_DBCCDB da una holgura de 10% para el crecimiento en el tamaño de la base de datos designado
- Por consiguiente, el tamaño del scan workspace se calcula como 1.2% del tamaño total de la base de datos

Text workspace

- El procedimiento SP_PLAN_DBCCDB calcula el text workspace como el 25% del tamaño de el scan workspace
- Ajustando el tamaño de text workspace
  - Para ambientes que usan alguno o ninguna columna del texto, usted puede hacer los workspace del texto más pequeño que la recomienda por SP_PLAN_DBCCDB
  - El tamaño mínimo, en páginas, para un workspace del texto es:
  - (número de worker process usados + 1) * 8
  - Para un base de datos de 1GB que no usa datos del tipo text/image haga el tamaño de workspace del texto al mínimo y ahorraría alrededor de 2.6MB

- Usted puede necesitar aumentar el tamaño del workspace del texto más allá que el SP_PLAN_DBCCDB recomendó
- La primera vez que el checkstorage se usa, despliega un mensaje si el tamaño del workspace del texto necesita ser aumentado

Planeando Recursos: Named Caches

- Cuando configure el tamaño del caché para dbccdb, use el valor recomendado de SP_PLAN_DBCCDB. Recuerde:
  - configurar el caché con SP_CACHECONFIG
  - configurar bancos de buffer para el caché usando SP_POOLCONFIG
  - vincular el caché a la base de datos dbccdb usando SP_BINDCACHE
- Durante una ejecución del checkstorage, los workspaces se ligan dinámicamente al cache configurado para la base de datos
- El cache usado debe configurarse con un pool de 16K
- El tamaño del cache recomendado por SP_PLAN_DBCCDB aplica a el buffer pool de 16K y es igual a :
  - El 20% del tamaño del workspace para la base de datos designada
  - 640K por worker process usado
- Asignando más memoria a el buffer pool de 16K que el recomendado por SP_PLAN_DBCCDB puede aumentar el performance de checkstorage
- El comando CHECKSTORAGE sólo usa el buffer pool de 16K en el cache asignado a él
- Si el cache usado es compartido por otras aplicaciones el chckstorage puede experimentar una degradación de la actuación mientras esta corriendo

Planeando Recursos: Worker Processes

- Considere los siguientes cuatro conceptos:
  - Ws: El número total de worker processes configurado
  - Wp: El número de worker processes sugerido por SP_PLAN_DBCCDB uno por dispositivo en el que la base de datos designada reside
  - Wc: El número de workers configurado para una base de datos particular
  - Wu: El número de workers realmente usado por checkstorage
- El número de worker processes configurado para el checkstorage no es el número necesariamente usado
- los checkstorage pueden usar menos worker processes que Wp o Wc

Fase 2: Inicializando la base de datos dbccdb

- Después de que la fase de planeación de recursos está completo, haga éstos los pasos de instalación:
  1. inicialice los dispositivos (optativo)
  2. cree la base de datos dbccdb
  3. configure el log de dbccdb (optativo)
  4. cree y mapee los segmentos de la base de datos (optativo)
  5. cree e inicialice las tablas del dbccdb
  6. cree el workspaces
  7. configure dbccdb para las bases de datos designado
- Paso 1: Inicialice los dispositivos usando el comando DISK INIT
  - Esté seguro usar dispositivos físicos particulares cuando sea posible
- Paso 2: Cree la base de datos dbccdb
  - Esté seguro poner los datos y el segmento de log para el dbccdb en dispositivos diferentes, para evitar problemas y performance bajo
- Paso 3: Configure la dbccdb para el manejo del log que usa uno de los métodos siguientes:
  - Habilite la opción truncate log on chkpt de la base de datos dbccdb
  - Agregue un umbral de la última oportunidad en dbccdb para que el log de transacciones se respalde automáticamente. El fracaso para realizar estos pasos puede producir que el log se llene y el checkstorage se suspende o es terminado por el usuario
- Paso 4: Cree y mapee segmentos de la base de datos para el workspaces
  - Este paso es optativo; sólo aplicable cuando se esperan que las bases de datos analizándose crecerán con el tiempo
  - Si los tamaños de la base de datos designada son estáticos, salte este paso
  - Use segmentos para asegurar crecimientos futuros del workspaces
- Paso 5: Cree e inicialice las tablas del dbccdb
  - El script $sybase/scripts/installdbccdb es proporcionado para:
    - Instalar las ocho tablas de control de dbcc (es necesario)
    - Instala los procedimientos del dbcc
    - Inicializa la tabla dbcc_types insertando 80 filas
  - Si una tabla ya existe, se borrar y sólo recrea si una tabla de la que depende no existe
- Paso 6: Cree el workspace

- Ø El procedimiento SP_DBCC_CREATEWS se usa para crear los dos scan y workspaces del texto.
- Ø Un nombre del segmento debe proporcionarse; si el usuario no definió segmentos, entonces el segmento predefinido debe especificarse

Ø Paso 7: Configure las bases de datos fuentes en dbccdb
- Ø Antes de que puedan correrse checkstorage, debe configurarse información sobre la base de datos a analizar explícitamente en la base de datos dbccdb
- Ø El procedimiento SP_DBCC_UPDATECONFIG es usado para configurar atributos de base de datos especificados en dbccdb. Más específicamente, para cada base de datos a ser checados debe ser configurada. DBCC CHECKSTORAGE no checa cualquier base de datos que no tenga esos valores en sus atributos
- Ø Pueden configurarse siete atributos diferentes cuatro obligatorios, deben ponerse de la base de datos a analizarse antes del checkstorage correrá contra él

Atributos de dbccdb

Nombre del Atributo	Requerido ? / Default	Descripción
max worker processes	Yes/NA	Max número de worker processes que puede ser usados
dbcc named cache	Yes/NA	Nombre y tamaño del cache ligado
scan workspace	Yes/NA	Nombre de scan workspace
text workspace	Yes/NA	Nombre de text workspace
OAM count threshold	No / 2%	Si la página de contadores en el OAM de un objeto difiere de la página de contadores real por más de este porcentaje, entonces los chequeos del objeto se descontinuará
IO error abort	No / 10	El número de errores de I/O permitido en un dispositivo particular antes del chequeo de objetos en ese dispositivo se discontinuará
linkage error abort	No / 10	El número de errores de la unión permitido en un objeto particular antes del chequeo de ese objeto se discontinuará

Configurando dbccdb para una base de datos fuente

Ø El procedimiento SP_DBCC_UPDATECONFIG debe ejecutarse cuatro veces por lo menos para configurar una base de datos analizada particular:

```
sp_dbcc_updateconfig db1, 'max worker processes', '2'
sp_dbcc_updateconfig db1,'dbcc named cache','cache1','8M'
sp_dbcc_updateconfig db1,'scan workspace', scan_ws
sp_dbcc_updateconfig db1,'text workspace', text_ws
```

Evaluando la configuración de la base de datos fuente

Ø El checkstorage graba información considerando:
- Ø Cualquier error que descubrió en la base de datos
- Ø Las estadísticas características de los datos en la base de datos designado

Ø El procedimiento SP_DBCC_EVALUATEDB usa este datos estadísticos para:
- Ø Reevaluar la configuración de una base de datos designada y recomendar valores más exactos para:
  - Ø Named cache size
  - Ø Scan workspace size
  - Ø Text workspace size
  - Ø Process count

Usando sp_dbcc_evaluatedb

Para debugear problemas de checkstorage

Ø Si los checkstorage no corren contra una base de datos particular, usted puede usar el procedimiento SP_DBCC_EVALUATEDB (en algunos casos) ayuda a analizar el problema
- Ø Los datos de configuración no existen
  - Ø Si el SP_DBCC_EVALUATEDB se corre contra una base de datos que no contiene ningún dato de configuración en la tabla dbccdb..dbcc_config, fallará con el error siguiente:

```
Msg 18474
Sales database is not configured for DBCC in the dbcc_config table. Use
sp_dbcc_updateconfig to configure it.
```

Ø En este escenario, los usuarios deben:
- Ø Ejecutar SP_PLAN_DBCCDB para conseguir estimaciones
- Ø Ejecute SP_DBCC_UPDATECONFIG para insertar los datos de configuración

- Si el sp_DBCC_EVALUATEDB se corre contra una base de datos con por lo menos una fila correspondiente en la tabla del dbcc_config:
  - Examina la configuración actual
  - Examina datos del checkstorage previamente ejecutado
  - Despliega la configuración actual
  - Despliega la configuración recomendada
- Para situaciones en las que los checkstorage no corren, use la actual y la configuración recomendada para poner a punto

Poniendo a punto Problemas de setup

- Incluido el bajo rendimiento de ejecución usar el SP_DBCC_EVALUATEDB contra una base de datos con extrañas opciones de configuración:

```
1> sp_dbcc_evaluatedb db1
2> go
 Recommended values for workspace size, cache size and process count are:
   Database name : db1
current scan workspace size : *OK (Indica ninguna entrada)
current text workspace size : OK (Indica ninguna entrada)
current cache size : OK (Indica ninguna entrada)
current process count : 2 (Solamente require la entrada proporcionada)
suggested scan workspace size : 48K
suggested text workspace size : 12K
suggested cache size : 640K
suggested process count : 1
```

Si desplegó valores actuales menores a valores sugeridos, los checkstorage no correrán
*0K (ceros K) indica ninguna entrada o una entrada de 0

Trampas potenciales de la Instalación

Los checkstorage no correrán

- Algunas situaciones impiden al checkstorage correr y producen errores:

Situación	Error
La base de datos dbccdb no encontrada	9964
La tabla dbcc no existe	208 & 9965
La config. de la tabla dbcc_config no es correcta	9965
El número de worker processes disponible es menor que el número configurado	9961
El scan workspace no existe	9966
El scan workspace existe, pero el ID del objeto no es el mismo en dbcc_config	9966
El scan workspace es demasiado pequeño	9952
El text workspace no existe	9967
El text workspace existe, Pero el ID del objeto no es el mismo del ID en dbcc_config	9967
El cache referido no existe	9977
El buffer pool de 16K en el cache no existe o es muy corto	9978

Arquitectura del dbcc checkstorage

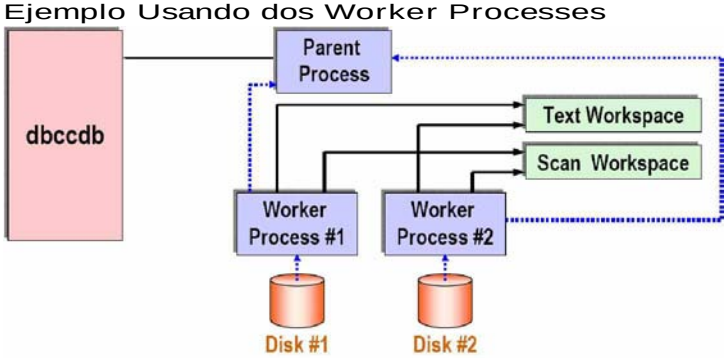
- La operación del checkstorage incluye 5 fases:
  - 1. Inicialización
  - 2. Búsqueda de base de datos
  - 3. Chequeos de ligado de páginas
  - 4. Chequeo de OAM
  - 5. Terminación
- Cada fase lee y escribe a la base de datos de dbcc
- Antes de discutir los detalles de cada fase, una vista previa rápida del esquema del dbcc base de datos es necesaria

Fase Database Scan: Apreciación global

Abajo se explica como DBCC CHECKSTORAGE se ejecuta:

1. Inicialización – Verifica la configuración de la base de datos destino y la disponibilidad de los recursos requeridos
2. Búsqueda en la base de datos – Lee toda la base de datos tan rápido como sea posible, realiza algunos chequeos y escribe un breve resumen de cada página del workspace en dbccdb
3. Los chequeos que quedan son realizados en dbccdb
4. Cuando una inconsistencia es encontrada no puede ser atribuida a la actividad de actualización concurrente, esto es registrado en dbccdb

Ejemplo Usando dos Worker Processes



Manejando Faults

Examinando resultados

- El primer paso en el manejo de Faults descubiertos por checkstorage es comprender por que ellos existen
- el checkstorage no escribe sus resultados a un archivo, graba sus resultados en la base de datos dbccdb
- Por consiguiente, se exigen a los usuarios que examinen los resultados del chequeo explícitamente
- el checkstorage hace, despliega un mensaje sumario a la sesión en la que fue invocado que incluye:
  - Número de la secuencial (opid) de la ejecución descrita
  - Número actual (hard) de Faults y Faults sospechosos
  - El número a nivel objeto de chequeos abortados

Resumen de mensajes del checkstorage

- El checkstorage típicamente despliega cualquiera de los siguientes dos mensajes a la sesión en la que se invocó:

Storage checks for 'ChrisDB' are complete. DBCC is now recording the results in the dbccdb database.

DBCC CHECKSTORAGE for database 'ChrisDB' sequence 1 completed at May 20 1997 8:29AM. 3 faults and 0 suspect conditions were located. 0 checks were aborted.You should investigate the recorded faults, and plan a course of action that will correct them.

Examinando Resultados

- Mecanismos para examinar los resultados del checkstorage:
  - Usando el procedimientos dbcc
  - Dependiendo de los requisitos específicos de un usuario y nivel especializado, considere hacer querys directamente a la tabla del dbccdb
  - Consulte el manejador dbcc de la base de datos usando triggers

Procedimientos del sistema DBCC

- Estos procedimientos se proporcionan con la base de datos dbccdb:
  - SP_DBCC_SUMMARYREPORT: Despliega un resumen de los datos como el número de errores duros y suaves informado contra la base de datos especificada para cada ejecución del checkstorage guardada en dbccdb
  - SP_DBCC_FAULTREPORT: Despliega el nombre del objeto, ID del índice, número de la página y descripción de cada error encontrado en la última ejecución del checkstorage
  - SP_DBCC_DIFFERENTIALREPORT: Compara valores de contadores desde dos casos de checkstorage en la base de datos dbccdb y reporta sólo valores que cambiaron (los contadores aplican a faltas y estadísticas)
  - SP_DBCC_STATISTICSREPORT: Despliega los valores del contador actual para cada objeto en la base de datos designado; la ejecución puede ser muy largo
  - SP_DBCC_CONFIGREPORT: Despliega la configuración actual para la base de datos designado; similar al SP_DBCC_EVALUATEDB
  - SP_DBCC_FULLREPORT: Despliega la configuración, estadísticas y datos de faltas para una base de datos dada o objeto

Usando dbcc Stored Procedures

SP_DBCC_SUMMARYREPORT

- Use este procedimiento para desplegar resultados sumarios sobre las bases de datos en dbccdb
- Este procedimiento despliega esta información para cada ejecución grabada de checkstorage; por consiguiente, el rendimiento puede abarcar mucho tiempo

```
 1> sp_dbcc_summaryreport
 2> go
Database     Date Start time End Time Hard Soft Text Abort Count
-------- -------- ---------- -------- ---- ---- ---- -----------
db1      05/02/97   10:01:47 10:02:25    1    4    0           0
db2      05/02/97   12:10:02 12:10:08    0    9    0           0
db2      05/02/97   12:45:11 12:45:16    2    7    0           1
```

Ø Use esta información para determinar rápidamente:
  Ø El número de faltas en una base de datos
  Ø Número de chequeos abortados (da el número de objeto con problemas en su unión de página)
  Ø Promedio del tiempo de la ejecución
  Ø Número de valores no-nulos tipo texto

SP_DBCC_FAULTREPORT: Reporte corto

Ø Este procedimiento despliega información adicional sobre las faltas particulares descubiertas por la más reciente ejecución de checkstorage

```
sp_dbcc_faultreport sybsystemprocs
go
Database Name: sybsystemprocs

Table Name     Index Type   Code         Description Page Number
-------------- ---------- ------ ------------------ -----------
sysprocedures           0 100031 page not allocated        5702
sysprocedures           1 100031 page not allocated       14151
syslogs                 0 100022 chain start error        24315
syslogs                 0 100031 page not allocated       24315
```

Ø Por defecto, este procedimiento se ejecuta en el modo corto, como el mostrado:
Ø Use esta información para determinar rápidamente:
  Ø El objects/indid específico con errores
  Ø Los números de la página específico involucrados
  Ø La descripción breve de la falta que descubrió
Ø Esta salida no indica qué faltas son duras y qué es suave

Ø El informe largo de este procedimiento se invoca vía la sintaxis siguiente:

```
sp_dbcc_faultreport long, [database name]
```

Ø Este reporte es más extenso:

```
sp_dbcc_faultreport long, jakedb
go
Generating 'Fault Report' for object mytab in database jakedb.
Fault Code: 100022; Hard fault
chain start reference disagrees with page linkage.
page id: 313
page header: 0x0000013900000000000013900F430100000056F00...
Header for 313, next 0, previous 313, id = 16003088:0
  TS = 0x00010000056F, next row = 0, level = 0
  free offset = 32, minlen = 70, status = 129(0x0081)

Generating 'Fault Report' for object mytab in database jakedb.
Fault Code: 100002; Soft fault, possibly spurious
page free offset value in header is invalid.
page id: 313
page header: 0x0000013900000000000013900F430100000056F00...
Header for 313, next 0, previous 313, id = 16003088:0
  TS = 0x00010000056F, next row = 0, level = 0
  free offset = 0, minlen = 70, status = 129(0x0081)
```

Ejecute sp_dbcc_evaluatedb

Ø Si las características de una base de datos de usuario cambia, use el procedimiento almacenado SP_DBCC_EVALUATEDB para reevaluar la configuración actual de dbccdb
Ø Los cambios posteriores a una base de datos de usuario debe afectar la configuración de dbccdb
Ø Cuando una base de datos de usuario es creada, borrada o alterada, el tamaño de los workspaces y del caché deben ser afectados
Ø Los cambios en el tamaño de el caché o los procesos worker cuentan para que DBCC_CHECKSTORAGE deba requerir una reconfiguración en el caché del buffer del SQL Server y los procesos worker

Restringiendo Recursos para evitar Dispendio

Porqué Fijar Límites en los Recursos?

Ø El cliente requiere tanto consultas aleatorias ad-hoc o consultas de aplicaciones erróneas que pueden tener un efecto adverso en el funcionamiento de todo el servidor. Las demandas pueden requerir un anormal número alto de recursos y que afectan negativamente el rendimiento de el servidor

Descripción. El Administrador de recursos provee una manera para que el administrador del sistema pueda poner límites en recursos.

Ø Definir que acción deberá ser tomada cuando un límite es violado
Ø Definir cuáles veces del día o semana los límites de recursos estarán en efecto
Ø Los límites de recursos pueden estar basados en logines individuales o aplicaciones
Ø El uso de esos límites permite al administrador del sistema prevenir consultas con fallas para reducir el funcionamiento del SQL Server
Ø El tipo de limites incluye:
  Ø Numero de I/O (estimado y actual)
  Ø Tiempo de Ejecución (de un bach o transacción)
  Ø Número de registros.
Ø Acciones para asumir cuando ocurra una violación al limite:
  Ø Mensaje de Advertencia y continua el proceso.
  Ø Abortar la consulta
  Ø Abortar la transacción
  Ø Desconectar la sesión del usuario
Ø Los límites del recurso pueden estar activos en momentos diferentes
Ø Tiene la habilidad de crear nombres para rangos de tiempo durante los cuales las restricciones están en efecto

Habilitando Límites de Recurso

Ø Debe configurarse para permitir límites de recurso
Ø Use la opción allow resource limits de la configuración:

```
1> sp_configure "allow resource limits",1
2> go
Parameter Name          Default Memory Used Config Value Run Value
---------------------- ------- ----------- ------------ ---------
allow resource limits        0           0            1         0

Configuration option changed. ASE must be rebooted before the change in
effect since the option is static.

(return status = 0)
```

Creando un Límite de Recurso

Ø Un límite de recurso se crea usando el procedimiento SP_ADD_RESOURCE_LIMIT. Sintaxis:

```
sp_add_resource_limit name, appname, rangename,
    limittype, limit_value, enforced, action, scope
```

Ø Crear un rango de tiempo con el procedimiento de sistema SP_ADD_TIME_RANGE
Ø Cuando se crean límites de recursos contra aplicaciones de los usuarios, use las llamadas Open Client apropiadas para nombrar la aplicación

Límite de Recurso (name y appname)

Name. El login al cual se le aplicara el límite de recurso
Appname. El nombre de la aplicación al que el límite de recurso aplica

Ø Uno de los dos parámetros puede ser NULL
  Ø Para crear un límite que aplica a todos los usuarios de una aplicación particular, especifique en name de NULL
  Ø Para crear un límite que aplica a todas las aplicaciones usada por un login, especifique un appname de NULL
Ø Ambos name y appname pueden ser no NULL
  Ø Para crear un límite que aplica a un login específico que usan una aplicación específica, indique ambos name y appname
Ø Por lo menos uno debe ser no NULL

1. Login como UserN (su nombre de usuario)
2. Agregar un límite de recurso en Tom42

```
sp_add_resource_limit Tom42, NULL, "at all times", row_count, 100,
                      2, 2, 1
```

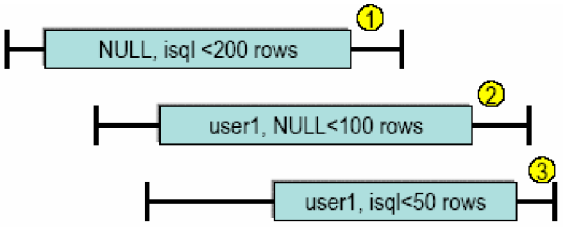
3. Para probar, salga de la sesión y vuelva a conectarse como Tom42

```
select * from pub2..intro1
```

4. Examine el error log con “more <logname> | tail -20”

Creando una Jerarquía de Límite de Recurso

Ø Una jerarquía de límite de recurso puede ser creada asignando límites separadamente a la aplicación y a login



Rango de Time para el Límite de recurso (rangename)

Ø Un límite de recurso activo es determinado a través de un rango de tiempo nombrado
Ø Es creado por el procedimiento SP_ADD_TIME_RANGE

```
sp_add_time_range name, startday, endday, starttime, endtime
```

Ø El rango de tiempo predefinido se llama at all times
  Ø Trabaja todo el tiempo, desde el primer día de la semana hasta el último, desde 00:00 a 23:59
  Ø No puede ser modificado o borrado
Ø Crea un time range llamado evenings, que esta activo de lunes a viernes de las 6 p.m. A medianoche

```
sp_add_time_range "evenings", "Monday", "Friday", "18:00", "00:00"
go
```

Tipos de Límite de recurso (limittype)

Ø I/O cost (io_cost)
  Ø showplan proporciona información que estima el costó I/O
  Ø statistics io proporciona información del costó real de I/O
Ø Tiempo transcurrido de ejecución (elapsed_time)
Ø El número de filas devueltos (row_count)

Valor de Límite de recurso (limit_value)

Ø El valor se espera como un valor del tipo integer
Ø El valor especifica:
  Ø I/O cost en suma total
  Ø Elapsed time en segundos
  Ø El numero de renglones
Ø Para conseguir tener idea de posibles valores limites use io_cost, ejecutando la opción SET STATISTICS IO o SET SHOWPLAN

Notas del limit_value

Ø Un límite en io_cost controla la cantidad de I/O que un query puede hacer. Este tipo del límite puede usarse para controlar I/O de querys intensivos, pero no controla querys simples que devuelven un juego del resultado grande.
Ø Un límite en row_count controla el número de filas devuelto a un usuario desde un query. No controla preguntas complejas que devuelven un resultado pequeño.
  Ø No incluya filas generadas en tablas temporales de temdb
Ø Un límite en elapsed_time controla la cantidad de tiempo que toma para devolverle resultados al usuario. Tenga presente que una pregunta puede tomar un tiempo largo debido a su complejidad, la carga del servidor, la espera por bloqueos y otros.

Momento del Limite de Recurso (enforced)

Ø Pueden forzarse los límites de recurso a priori o durante la ejecución del query. La siguiente tabla lista los valores válidos por

Enforced	Descripción	Tipo de Limite
1	La acción se toma cuando el costo estimado o el costo real excede el límite especificado	Io_cost
2	La acción se toma cuando el contador de fila real, tiempo transcurrido o el costo de ejecución excede el límite especificado.	Row_count Elapsed_time Io_cost
3	La acción se toma cuando el costo estimado o el costo real excede el limite especificado	Io_cost

Acciones a tomar cuando hay Violación al limite (action)

Ø Especifica la acción para tomar cuando el límite se excede

Ø Los códigos de acción siguientes son válidos para todos los tipos del límite:

Acción	Descripción
1	Emite una advertencia
2	Aborta el Quero batch.
3	Aborta la transacción
4	Elimina la sesión

Ejemplo Límite de Recurso (row_count)

Cree un límite de recurso que es activo at all times (default) para prevenir que aplicación isql no retorne más de 200 renglones; la violación se produce terminando el batch

```
 1> sp_add_resource_limit NULL,'isql','at all times','row_count', 200,
 2> 2, 2, 1
 3> go
  New resource limit created.
  (return status = 0)
```

Violación del recurso: Perspectiva del usuario

Ø Perspectiva del usuario:

```
  1> select * from test_table
  2> exec sp_who
  3> go
  . . .
  200 Testing
  Row count exceeded limit of 200.
  Command batch has been aborted.
```

Violación del recurso: Perspectiva del SA

Ø Perspectiva del SA de la violación del recurso (errorlog). Se manda el mensaje al error log:

```
$ tail $SYBASE/install/errorlog_SYBASE
 00:0000:96/08/02 15:14:53.27 server User
 'user1', application 'isql' exceeded row count limit of 200.
```

Ejemplo Límite de Recurso (elapsed_time)

Ø Cree un límite de recurso que es activo at all times (default) para impedir al isql pasar más de 120 segundos para cada transacción; la violación produce el aborte de la transacción

```
  1> sp_add_resource_limit NULL,'isql','at all times','elapsed_time',
  2> 120, 2, 3, 4
  3> go
  New resource limit created.
```

Ø Pruebe el límite de recurso

```
  1> begin tran
  2> insert into test_table values (1, "Insert")
  3> go
/*Waiting 120 seconds*/ Elapsed time exceeded limit of 120.
Transaction has been aborted.
```

La tabla de Sistema spt_limit_types

Ø La tabla spt_limit_types mantiene los tipos de límite

```
  1> Select * from master.dbo.spt_limit_types
  2> go
name          id enforced object_type scope units
------------ -- -------- ----------- ----------
io_cost        1        3           1           1
derived from optimizer's costing formula

elapsed_time   2        2           1           6
in seconds

row_count      3        2           1           1
# of row returned to client

(3 rows affected)
```

La tabla del sistema sysresourcelimits

Ø Ejemplo:

```
  1> select * from master.dbo.sysresourcelimits
  2> go
```

```
name  appname ranged limitd enforced action limt value scopespare
----- ------- ------ ------ -------- ------ ---- ----- ----------
user1 NULL          2      3        2      2   50     1          0

(1 row affected)
```

La tabla de Sistema systimeranges

Ø La tabla systimeranges mantiene todos los rangos de tiempo conocidos
Ø El rango de tiempo predefinido es el de "at all times". Ejemplo:

```
 1> select * from systimeranges
 2> go
name           id  startday  enday  starttime  endtime
------------   --  --------  -----  ---------  -------
at all times    1         1      7      00:00    00:00

(1 row affected)
```

Notas para Crear Límites de Recurso

Ø Un límite se liga a la sesión del usuario en momento de entrar a sesión
Ø Se informan errores para los nombres del login inválidos, el rango de tiempo o los tipos del límite no encontrados en syslogins, systimeranges y spt_limit_types
Ø Los limitvalue deben ser positivos
Ø Solamente pueden limitarse los io_cost antes de la ejecución
  Ø los tipos row_count y elapsed_time limitan durante la ejecución
Ø El aborte del lote es la acción predefinida de la violación para todos los límites.

Modificando Límites de Recurso

Ø Modifique los valores del recurso de límite usando SP_MODIFY_RESOURCE_LIMIT:
Ø sp_modify_resource_limit name , appname, rangename, limittype
  [, limitvalue] [, enforced] [, action] [, scope]
Ø Pueden modificarse sólo valor del límite y acción a tomar
  Ø Borre y recree el límite si requieren modificación otros parámetros

Ejemplo de Modificación de Límite de Recurso

Ø Cambie limitvalue de 200 a 50:

```
 1> sp_modify_resource_limit NULL, 'isql','at all times', 'row_count',
    50, 2, 2, 1
 3> go
  (return status = 0)
```

Pruebe el límite del recurso:

```
 1> select * from test_table
 2> go
  50 Testing
  Row count exceeded limit of 50.
  Command batch has been aborted.
```

Consiguiendo Información Sobre los Límites de Recurso

Ø Use el procedimiento de sistema SP_HELP_RESOURCE_LIMIT para determinar qué límites del recurso aplican a un usuario dado, aplicación o tiempo del día. Sintaxis:

```
sp_help_resource_limit [name [, appname [, limittime [,limitday
                [,scope [, action]]]]]]
```

Ø Ejemplos:

```
sp_help_resource_limit NULL, my_app
sp_help_resource_limit NULL, NULL, "09:00"
sp_help_resource_limit joe_user, NULL, NULL, Sunday
```

Borrando Límites de Recurso(s)

Ø Para borra uno o más recurso use el procedimiento SP_DROP_RESOURCE_LIMIT (debe ser sa). Sintaxis:

```
sp_drop_resource_limit {name, appname} [,rangename, limittype, enforced,
action, scope]
```

Ø Name. Se fuerza al login para que el limite aplique
  Ø Especifique a un usuario para borrar límites de recurso de un login
  Ø Para Borrar límites de recurso que aplican a todos los usuarios de una aplicación en particular, especifique un nombre de NULL
Ø Appname. La aplicación a la que el límite aplica
  Ø Especifique un nombre de la aplicación para borrar los límites de recurso de esa aplicación
  Ø Para borra límites de recurso que aplican a todas las aplicaciones usados por el login especificado, especifique un appname de NULL
Ø Rangename. El rango de tiempo durante el que el límite trabaja

- Ø Éste debe ser un rango de tiempo existiendo o NULL para anular todos los límites del recurso para el nombre especificado, appname, limittype, acción y alcance, sin tener en cuenta el rangename.

Ø Limittype. El tipo de recurso que está limitado

Tipo de Limite

Limite	Descripción
row_count	Borra sólo límites que restringen el número de filas que un query pueden devolver.
elapsed_time	Borra sólo límites que restringen el número de segundos que un query batch o la transacción puede correr
io_cost	Borra sólo límites de costo que restringen real o estimado del query que procesa.
NULL	Borra todos los límites del recurso con el nombre especificado, appname, rangename, acción y alcance, sin tener en cuenta el limittype.

action – La acción tomada cuando el límite se

Acción	Descripción
1	Borra sólo límites que emiten una advertencia.
2	Borra sólo límites que abortan el query batch
3	Borra sólo límites que abortan la transaction.
4	Borra sólo límites que eliminan la sesión
NULL	Borra todos los límites de recurso con el nombre especificado, appname, rangename, limittype y alcance, sin tener en cuenta la acción que toman.

scope – El alcance del límite

Scope	Descripción
1	Borra sólo límites a los que aplican a queries
2	Borra sólo límites a los que aplican a query batches
3	Borra sólo límites a los que aplican a transactions
4	Borra sólo límites a los que aplican a ambos query batches y transactions
NULL	Borra todos los límites de recurso con el nombre especificado, appname, rangename, limittype y acción, sin tener en cuenta su alcance

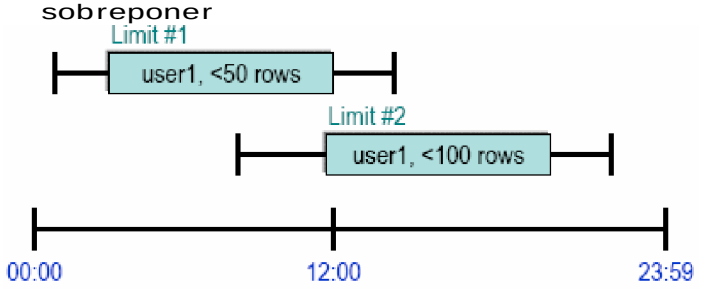
Borrando Límites de Recurso(s) ejemplos

1. Borra un solo límite del recurso que mata la sesión siempre que joe usa la aplicación de nómina los viernes por la tarde y tienen resultados en costo de I/O excesivo:
sp_drop_resource_limit joe, payroll, fridy_aftern, io_cost, NULL, 4, 1
2. Borra todos los límites que aplican al uso de joe de la aplicación de la nómina:
sp_drop_resource_limit joe, payroll, NULL, NULL, NULL, NULL, NULL
3. Borra todos los límites que aplican al usuario joe
sp_drop_resource_limit joe, NULL, NULL, NULL, NULL, NULL, NULL
4. Borra todos los límites de recurso que aplican a la aplicación de la nómina:
sp_drop_resource_limit NULL, payroll, NULL, NULL, NULL, NULL, NULL
5. Borra todos los límites de recurso en la aplicación de la nómina cuya acción es matar la sesión:
sp_drop_resource_limit NULL, payroll, NULL, NULL, 4, NULL, NULL

Ø Cuando usted usa SP_DROPLOGIN para borrar un login, se borran todos los límites de recurso asociados con ese login automáticamente

ERROR: El rango de tiempo que se sobreponen

Ø No, el recurso limita por usuario, nombre de la aplicación, el limittype no se puede sobreponer



Impacto del recurso

Rendimiento

- El administrador de recurso causa menos del 1% de impacto de el performance del servidor en medidas de transacciones por segundo (TPC-B y TPC-C)

Memoria

- El administrador de recurso requiere algunas estructuras de memoria interiores adicionales
- Cambios después de permitir límites de recurso:
  - el cache del procedimiento disminuyó de 1028 a 1026 (KB)
  - el número de alarmas aumentó de 2 a 4 (KB)

Acceso a Datos a Través de la Programación de Clientes

Definiciones

ODBC. Open Database Connectivity, Método estándar de acceso a bases de datos desarrollado por Microsoft. Su propósito es hacer posible el acceso de cualquier dato, gestionado por un RDBMS, desde cualquier aplicación.

JDBC

"Java Database Connectivity" (Conectividad de Base de Datos de Java), es un marco de programación para los desarrolladores Java que escriben programas que tienen acceso a la información guardada en bases de datos. JDBC se utiliza comúnmente para conectar un programa del usuario con una base de datos por "detrás de la escena", sin importar qué software de administración o manejo de base de datos se utilice para controlarlo. Esto se logra utilizando las clases JDBC API, que está disponible en el sitio de Sun.

```
Connection getConnection() {
try {Class.forName("com.sybase.jdbc.SybDriver");
con = DriverManager.getConnection("jdbc:sybase:Tds:teseo.dcaa.unam.mx:
4100/Base","Usuario","password");
} catch (SQLException sqle) {sqle.printStackTrace();
} catch (ClassNotFoundException cnfe) {cnfe.printStackTrace();
}
return con;
}
```

PHP

Hypertext Preprocessor. Lenguaje de programación de distribución libre que permite realizar scripts con código HTML "embebido" para crear páginas web dinámicas. Ofrece acceso a RDBMS's por medio de la compilación de librerías específicas con APACHE.

```
<?
class conexion Pg {
var $conexion;
function getConexion() {
$this->conexion=@pg_connect("host=localhost port=5432 user=cPhp
password=curs0 dbname=Agenda");
if($this->conexion) {
#echo "Se obtuvo la conexion";
return $this->conexion;
}
else {echo "No Hay Conexión";}
}
}
?>
```

Cliente. Es cualquier aplicación que corre o se ejecuta en cualquier computadora personal, workstation y se comunica con un equipo servidor para ejecutar alguna operación especifica. Por ejemplo: acceso a bases de datos, accesos a correo electrónico, acceso a FTP, etc.

WWW

WWW. World Wide Web. En la web es utilizada la arquitectura denominada cliente – servidor en la que en una máquina central (servidor) se encuentran varios servicios a los que una o varias computadoras (cliente) pueden acceder, dependiendo de los privilegios que el usuario tenga dados de alta el servidor podrá tener acceso a ciertos servicios.

La relación entre el navegador del cliente y el servidor al ingresar a una página de hipertexto estática: El navegador del cliente hace la petición de un archivo HTML.

HTML

HTML
Hyper Text Markup Language. Es un sistema para estructurar documentos. Estos documentos pueden ser mostrados por los navegadores de páginas Web en Internet, como Netscape, Mozilla, Galeon, Microsoft Explorer, etcétera.

Cualquier cosa que hagamos debe ir entre etiquetas.
<ETIQUETA parámetros> ... </ETIQUETA>

Existen etiquetas que no necesitan cerrarse, así como etiquetas que no requieren de parámetros, más adelante se verán algunos ejemplos.

Un documento escrito en HTML contendría básicamente lo siguiente:

```
<HTML>  </HTML>  Indica el inicio y fin del documento.
<HEAD>  </HEAD>  Indica el inicio y fin del encabezado del documento.
<TITLE> </TITLE> Indica el inicio y fin del título del documento.
<BODY>  </BODY>  Indica el inicio y fin del cuerpo del documento.
```

Encabezado. La etiqueta <HEAD> </HEAD> delimita el encabezado, la etiqueta más importante dentro de ésta es <TITLE> en la cual se coloca el título de la página.

```
<HEAD>
<TITLE>Mi primera p&aacute;gina en html
</TITLE>
</HEAD>
```

Cuerpo. La etiqueta <BODY> </BODY> indica el contenido de nuestra página. Entre esta etiqueta pondremos texto, gráficos, ligas, etcétera. Tiene una serie de parámetros opcionales que permiten definir la apariencia general del documento:

- Background. Indica el nombre de una imagen que servirá como "fondo" de nuestra página. Si la imagen no rellena todo el fondo del documento, esta será reproducida tantas veces como sea necesario. Sintaxis: Background = "nombre de imagen".
- bgcolor. Indica un color para el fondo de nuestro documento. Se ignora si se usó el parámetro background. Sintaxis: bgcolor = "código de color"
- text. Indica un color para el texto que incluyamos en nuestro documento. Por defecto es negro. Sintaxis: text = "código de color".
- Link. Indica el color de los textos que dan acceso a una liga. Por defecto es azul. Sintaxis: link = "código de color"
- Vlink. Indica el color de los textos que dan acceso a una liga que ya hemos visitado. Por defecto es morado. Sintaxis = vlink = "código de color"

El código de color. Es un número compuesto por tres pares de cifras hexadecimales que indican la proporción de los colores rojo, verde y azul. El código de color se antecede del símbolo #. Ejemplo:

#000000 Color Negro
#FFFFFF Color Blanco

Etiquetas para dar formato a un párrafo. <P> </P> define las características de un párrafo en específico. Se puede usar el parámetro:

- Align. Sintaxis: Align = “LEFT | RIGHT | CENTER”
- <P> Indica un cambio de párrafo y deja una línea en blanco en medio.
-
 Sirve para indicar un salto de línea
- <HR> Inserta una línea horizontal en la página.

<A> </A> Permite definir ligas. el texto o imagen que este dentro de esta etiqueta será sensible, cuando se presione el botón izquierdo del ratón sobre esta sección se ejecutará lo que esta definido en el parámetro href de la etiqueta.

<!-- --> Comentario. Todo lo que este dentro de esta etiqueta es un comentario que no será mostrado en la página.

Cambiar el font. Las etiquetes <FONT> </FONT> permite variar el tamaño, el color y el tipo de letra de un texto determinado.
- Size. Da al texto un tamaño determinado. Sintaxis size = “valor”.
- Color. Escribe el texto en el color cuyo código se especifica. Sintaxis: color = "código de color"
- Face. Escribe el texto en el tipo de letra especificado. Si este tipo de letra no existe en la computadora que "lee“ la página se usará el tipo de letra predeterminado del navegador. Sintaxis: face = "nombre de tipo de letra"

Caracteres Especiales. Se utiliza para escribir caracteres que no son comunes en el idioma ingles
- ¿. & iquest. Signo de interrogación inicial
- ñ. & ntilde. La tilde de la letra ñ
- ‘. & acute. El apostrofe de acentuación de las vocales.

Tablas. Son elementos muy útiles para ordenar los contenidos de nuestras páginas. Se definen mediante la etiqueta <TABLE> </TABLE>, los parámetros opcionales son:
- border. Indica el ancho del borde de la tabla. Sinyaxis border =“número”
- cellspacing. Indica el espacio en pixeles que separa las celdas que están dentro de la tabla. Sintaxis: cellspacing =“número”.
- Cellpadding. Indica el espacio en pixeles que separa el borde de cada celda y el contenido de esta. Sintaxis: cellpadding = “número”
- Width. Indica el ancho de la tabla en pixeles o en porcentaje en función del ancho de la ventana del navegador. Si no se indica este parámetro, el ancho se adecuará al tamaño de los contenidos de las celdas. Sintaxis: width = “número o porcentaje”
- height Indica la altura de la tabla en pixeles o en porcentaje en función del alto de la ventana del navegador. Sintaxis: height = “número o porcentaje”
- bicolor. “código de color”. Especifica el color de fondo de toda la Tabla. Sintaxis: bgcolor = “código de color”

Encabezado, detalle y renglones de la Tabla. Para definir las celdas que componen la tabla se utilizan las directivas <TD> y <TH>. <TD> indica el detalle y <TH> indica el “título", El contenido será resaltado en negrita y en un tamaño ligeramente superior al normal. Para indicar una fila de la tabla se utiliza la etiqueta <TR></TR>. Sus parámetros opcionales son:
- Align. Indica como se debe alinear el contenido de la celda, a la izquierda, centrado, a la derecha o justificado. Sintaxis: align = “LEFT / CENTER / RIGHT / JUSTIFY”
- valign. Indica la alineación vertical del contenido de la celda, en la parte superior, en el centro o en la inferior. Sintaxis: valign = “TOP / MIDDLE / BOTTOM”.
- Rowspan. Indica el número de filas que ocupará la celda. Por defecto ocupa una sola fila. Sintaxis: rowspan = “número”.
- Colspan. Indica el número de columnas que ocupará la celda. Por defecto ocupa una sola columna. Sintaxis: colspan = “número”.
- Width. Indica el ancho de la columna en pixeles o en porcentaje en función del ancho de la ventana del navegador. Si no se indica este parámetro, el ancho se adecuará al tamaño de los contenidos. Sintaxis: width = “número o porcentaje”
- Bgcolor. Especifica el color de fondo del elemento de la Tabla. Sinatxis: bgcolor = “código de color”.

```
<TABLE border=4 cellspacing=4 cellpadding=4 width =80%>
<TH align=center>Nombre </TH>
<TH align=center colspan=2>Apellidos </TH>
<TR>
<TD align=LEFT>Ana </TD>
<TD align = LEFT>García </TD>
<TD align = LEFT>Hernández </TD>
</TR>
<TR>
<TD align = LEFT>José </TD>
<TD align = LEFT>Martínez </TD>
<TD align = LEFT>Cruz </TD>
</TR></TABLE>
```

Formularios

Permiten dentro de una página solicitar información al visitante y procesarla, se pueden solicitar diferentes datos cada uno de los cuales quedará asociado a una variable. Una vez se hayan introducido los valores en los campos, el contenido de estos será enviado a la dirección donde tengamos el programa que pueda procesar las variables. Este programa deberá estar escrito en un lenguaje de programación como ASP, Perl, PHP, etcétera.

La etiqueta <FORM> </FORM> permite definir formularios, sus parámetros son:

- Ø name. Indica el nombre de la forma. Sintaxis: name="nombre de la forma"
- Ø action. Indica la dirección del programa que va a procesar los datos recibidos de la forma. Sintaxis: action="programa"
- Ø method. Indica el método de transmisión a usar, POST para que los datos viajen ocultos y GET que si permite que se vean los valores y nombres de las variables en la barra de direcciones del navegador, este último es el valor por defecto. method="POST / GET".

Ya que todos los elementos de una forma serán recibidos por un programa que va a procesarlos deberán tener obligatoriamente el parámetro: name="nombre del elemento"

Cajas de Texto. Para definirlas se utiliza la etiqueta <INPUT> con el parámetro type="text", otros parámetros opcionales son:

- Ø maxlenght. Número máximo de caracteres a introducir en el campo. Sintaxis: maxlenght = "número"
- Ø size. Tamaño en caracteres que se mostrará en pantalla. Sintaxis: size = "numero"
- Ø value. Valor inicial del campo. Normalmente vacío. Sintaxis: value = "Valor".

Cajas de texto para introducir contraseña. Para definirlas se utiliza la etiqueta <INPUT> con el parámetro type="password" Mostrará asteriscos (*) en lugar de las letras escritas. Sus parámetros opcionales son los mismos que para las cajas de texto normales.

Cuadros de selección múltiple. Se utiliza la etiqueta <INPUT> con el parámetro type="checkbox" para definirlos. Permite elegir varios cuadros o casillas. Los valores de las casillas serán indicados por value="valor"

- Ø Checked. Parámetro opcional con el que la casilla aparecerá marcada por defecto.

Cuadros de selección sencilla. Se usa la etiqueta <INPUT> con el parámetro type="radio" para definirlos. Únicamente permite elegir un valor. Se utiliza value="valor" para definir los valores de las casillas.

Botones. Es el que dispara una acción. Tiene las siguientes opciones:

- Ø Value. Indica el texto que aparecerá en el botón. Sintaxis: value = "texto"
- Ø Type. La acción a tomar depende del valor de este parámetro. Sintaxis: Type = Submit | Reset
  - Ø Si type es "submit" todos los campos se envía al programa indicado en el parámetro action de <FORM>. Si type es "reset" se borra el contenido de todos los campos.

La etiqueta <INPUT> deberá tener el parámetro type="hidden". Este campo no será visible para el usuario. El valor se indica con el parámetro value="valor"

Listas de Selección. Despliegan una lista de opciones entre las que se pueden escoger una o varias. La etiqueta que se usa es <SELECT></SELECT> y sus parámetros son:

- Ø Size. Número de opciones visibles. Si se indica 1 se presenta como un menú desplegable, si se indica más de uno se presenta como una lista con barra de desplazamiento. Sintaxis: size="número"
- Ø Multiple. Permite seleccionar más de un valor para el campo.

Las diferentes opciones de la lista se indican con la etiqueta <OPTION>. Esta etiqueta puede incluir el parámetro selected para indicar cual es la opción por defecto. En caso de que no se especifique, se tomará por defecto la primera opción de la lista.

Áreas de texto. Es un campo de texto que permite incluir varias líneas, para definirlo se usa la etiqueta <TEXTAREA> </TEXTAREA>, sus parámetros son:

- Ø cols. Número de columnas de texto visibles. Sintaxis: cols="número"
- Ø rows. Número de filas de texto visibles. Sintaxis: rows="número"
- Ø wrap. Justifica el texto automáticamente en el interior de la caja. La opción PHYSICAL envía las líneas de texto separadas en líneas físicas. La opción VIRTUAL envía todo el texto seguido. Wrap ="VIRTUAL / PHYSICAL"

PHP Hipertext Preprocesor

PHP
Hipertext Preprocesor. Se combina con código HTML para generar páginas web dinámicas. Es ejecutado en el servidor y el resultado se envía al navegador, como se muestra en la siguiente figura.

Para indicarle al navegador que vamos a usar PHP debemos usar las siguientes etiquetas <?php y ?> o <? y ?>, todo lo que este entre esas etiquetas será interpretado como código PHP.

Usando un editor de texto cualquiera podemos crear nuestros scripts de PHP, dependiendo de la configuración del servidor en donde se alojarán los programas la extensión de los archivos podrá ser .php, .html o .phtml. Al final de cada instrucción deberá existir un punto y coma (;).

Para agregar comentarios en este lenguaje es igual que en C o C++:

```
/*Todo un texto comentado entre estos caracteres*/
//Un renglón comentado
# Otro renglón Comentado
```

Una variable en este lenguaje define su tipo según su contenido, las variables son de tipo escalar y tomarán un tipo determinado dependiendo de la asignación que se haga, podrán ser booleanas, enteras, de punto flotante o cadenas. En éste lenguaje también contamos con arreglos y objetos.

Es importante tomar en cuenta que PHP es sensible a mayúsculas y minúsculas, además de que no debe hacerse una declaración de variables previa ya que las variables se van generando conforme se van necesitando. Las variables se identifican porquen llevan el símbolo $ como prefijo.

Estructuras de control en PHP

IF
Permite la ejecución condicional de fragmentos de código.

```
if (expr)
{sentencia}
else
{sentencia)
```

expr se evalúa a su valor condicional. Si expr se evalúa como VERDADERO, se ejecutará la siguiente sentencia y posteriormente a la sentencia posterior a la sentencia del else y si se evalúa como FALSO se ejecuta la sentencia contenida en else.

Si se requiere ejecutar más de una sentencia cuando se dé cierta condición se deberá poner entre llaves.

```
if ($a > $b)
{
 print "a es mayor que b";
 $b = $a;
}
else
{print "a NO es mayor que b";    }
```

ELSEIF
La sentencia elseif se ejecuta sólo si la expresión if precedente y cualquier expresión elseif precedente se evalúan como FALSO y la expresión elseif actual se evalúa como VERDADERO.

WHILE
Su sintaxis es: while (expr) sentencia. Se ejecuta(n) la(s) sentencia(s) anidada(s) repetidamente, mientras la expresión while se evalúe como VERDADERO. El valor de la expresión es comprobado cada vez al principio del ciclo, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración. Si la expresión while se evalúa como FALSO desde el principio de todo, la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

```
while ($a < $b)
 { print "a es menor que b"; $a++; }
```

DO ... WHILE
Son muy similares a los ciclos while a diferencia que primero se ejecutan las sentencias y al final se evalúa la condición. La sentencia siempre se ejecutará al menos una vez.

```
do
{
sentencia
} while (expr)
```

FOR
Su sintaxis es: for (expr1; expr2; expr3). expr1 se evalúa incondicionalmente una vez al principio del ciclo. Al comienzo de cada iteración, se evalúa expr2 . Si se evalúa como VERDADERO, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como FALSO, la ejecución del bucle finaliza. Al final de cada iteración, se evalúa (ejecuta) expr3. Ejemplo:

```
for ($i = 1; $i <= 10; $i++)
 { print $i;}
```

SWITCH
Es similar a una serie de sentencias if en la misma expresión. En algunas ocasiones se requiere comparar una misma variable con valores diferentes y ejecutar diferentes sentencias dependiendo del valor de la variable. Sintaxis:

```
switch (variable)
 case valor1:
 sentencias;
 case valor2:
 sentencias;
 break;
```

BREAK
break escapa de las estructuras de control for, while o switch. Break acepta un parámetro opcional, el cual determina de cuantas estructuras de control hay que escapar.

Interrealación con Sybase

Para utilizar PHP con Sybase es necesario tener acceso a una cuenta de usuario que tenga instalado PHP y una cuenta con acceso a Sybase, puede ser la misma pero se deberá tener acceso tanto al lenguaje de programación como al manejador de base de datos.

Variables de configuración.
sybct.allow_persistent booleano. Permite conexiones persistentes con Sybase. El valor por default es "on".
sybct.max_persistent entero. Número máximo de conexiones persistentes por proceso, el valor por default es -1, esto significa que es ilimitado.
sybct.max_links entero. Número máximo de conexiones por proceso, incluyendo conexiones persistentes. El valor por default es -1 y significa que no hay un límite de conexiones.
sybct.hostname cadena. Nombre del Servidor.

Funciones que facilitan la interacción con Sybase.

- sybase_affected_rows ([int conexión]). Trae el número de columnas afectadas en la última consulta. Únicamente para INSERT, DELETE y UPDATE.
- sybase_close bool (int conexión). Cierra una conexión con Sybase. Regresa verdadero en caso de que sea exitoso. No cierra conexiones persistentes.
- sybase_connect int (nombre_servidor, nombre_usuario, password [,string charset]). Abre una conexión al servidor Sybase. Devuelve FALSE en error.
- sybase_data_seek bool (int variable_resultado_consulta, int número_registro). Coloca en una determinada posición el cursor de la consulta. Mueve el puntero interno asociado con el resultado de la consulta al número de registro especificado, la siguiente llamada a sybase_fetch_row() regresará ese registro. Regresa: TRUE en éxito, FALSE en falla.

- sybase_fetch_array array (int variable_consulta). Pone el resultado de una consulta en un arreglo. Devuelve un arreglo que corresponde al registro o FALSE si no hay más registros. Es una extensión a la versión de sybase_fetch_row().
- sybase_fetch_field object (int variable_resultado [, int campo_offset]). Trae la información del campo, regresa un objeto con dicha información. Puede ser usado para obtener información acerca de los campos en cierto resultado de una consulta. Las propiedades del objeto son:
  - name. Nombre de la columna, si la columna es el resultado de una función, esta propiedad es puesta en #N, donde #N es un número serial.
  - column_source. La tabla de la cual fue tomada la columna.
  - max_length. Tamaño máximo de la columna.
  - numeric. 1 si la columna es numérica.
  - type. Tipo de dato de la columna.
- sybase_fetch_object int (int variable_resultado). Trae un registro como objeto. Devuelve un objeto con propiedades que corresponden al registro o FALSE si no hay más registros. Es similar a sybase_fetch_array(), con una diferencia se regresa un objeto en lugar de un arreglo. Únicamente se pueden acceder a los datos por medio de los nombres de los campos y no por sus offsets.
- sybase_fetch_row array(int variable_resultado). Toma el registro como un arreglo enumerado. Trae un renglón de datos del resultado asociado con el identificador de resultado. El registro es devuelto como un arreglo. Cada columna del resultado es almacenada es un arreglo, el arreglo comienza en 0. La siguiente llamada a sybase_fetch_row() regresará el siguiente renglón en el resultado o FALSE si no hay más columnas.
- sybase_field_seek int (int variable_resultado, int campo_salida). Se coloca en un campo específico, si la siguiente llamada a sybase_fetch_field() no incluye el campo de salida, el anterior será regresado.
- sybase_free_result bool (int variable_resultado). Libera la memoria, únicamente se tiene que llamar si se esta preocupado de usar mucha memoria mientras el programa esta corriendo. La memoria será liberada cuando el programa termine.
- sybase_get_last_message string (void). Devuelve el último mensaje del servidor.
- sybase_num_fields int (int variable_resultado). Da el número de campos en el resultado.
- sybase_num_rows int (int variable_resultado). Da el número de renglones en un resultado.
- sybase_pconnect int (nombre_servidor, usuario, password [,string charset]). Abre una conexión persistente. Devuelve un identificador de conexión o FALSE si hay error. Las diferencias entre sybase_pconnect() y sybase_connect() son que cuando se conecta la primera trata de encontrar una conexión persistente con el mismo servidor, usuario y password, si es encontrada va a regresar ese valor en lugar de abrir otra conexión. Otra diferencia es que la conexión al servidor SQL no se va a cerrar cuando el programa termine, la conexión queda abierta para un uso futuro.
- sybase_query int (string query, int identificador). Manda una consulta al servidor que esta asociado con el identificador de conexión, si el identificador no ha sido especificado, el último identificador abierto es tomado. Si no existe una conexión abierta la función trata de establecer un identificador como si sybase_connect() hubiese sido llamado y lo usa. Devuelve el identificador si tuvo éxito, regresa FALSE si hay error.
- sybase_result string (int variable_resultado, int renglón, mixed campos). Da el resultado de una consulta. Devuelve el resultado de una celda en un registro. El argumento campo puede ser el nombre del campo, la tabla haciendo referencia al campo (tabla.campo). Si el nombre de la columna tiene un alias debe usarse el alias en lugar del nombre de la columna. Cuando se trabaja con largos resultados, se debe considerar usar una de las funciones que devuelven un solo registro. Estas funciones regresan el contenido de múltiples celdas en una llamada a la función, son más rápidas que sybase_result().
- sybase_select_db bool (string nombre_bd, int identificador_conexión). Selecciona la base de datos a usar. Regresa TRUE cuando hay éxito y FALSE cuando hay un error. Si no hay un identificador de conexión especificado, se toma el último por omisión. Si no hay una conexión abierta la función trata de establecer una como si sybase_connect() fuese llamada para usarse. Las siguientes llamadas a sybase_query() serán para la base de datos activa.

EJEMPLO

A continuación se encuentra un ejemplo que muestra como insertar en la base de datos de una agenda. Primeramente se necesita poner todos las datos para realizar la conexión a la base de datos este archivo tendrá el nombre de conexión.php y deberá estar incluido en todos los archivos que requieran conectarse a la base de datos. El archivo contendrá lo siguiente:

```
<?php
$conexion=sybase_connect ("servidor", "usuario", "password");
sybase_select_db("base_de_datos",$conexion);
?>
```

Se deberá contar con una forma html en la que podamos introducir los datos de los nuevos miembros de la agenda.

```
 <?php
 #Archivo que contiene los datos para la conexión a la base de datos.
 include ("./conexion.php");
 ?>

 <html><head><title>Registro</title></head>
 <body bgcolor="#FFFFFF">
 <center>REGISTRO</center>
 <!--Aquí se escribe a que archivo se mandan los datos de la forma.-->
 <form name="forma" method="post" action = "registro2.php">
 <table width="100" border="1" cellpadding="0" cellspacing="0"
class="fuente" align="center">
 <tr valign="middle" align="left">
<td align="left" valign="middle" width="76%" height="12"> <font
face="Verdana, Arial, Helvetica, sans-serif" size="1">
 <select name="pais" size="1">

 <?php
 #Realiza una consulta a la base de datos a la tabla PAIS
 $select = "SELECT * FROM PAIS ORDER BY NOMBRE";
 $query = sybase_query($select, $conexion);
 $num_ren = sybase_num_rows($query);
 for ($i=0;$i<$num_ren;$i++)
  {
  #Toma los valores que fueron devueltos en la consulta.
  $id_p = sybase_result ($query, $i, "ID");
  $nombre_p = sybase_result ($query, $i, "NOMBRE");
  #Imprime valores devueltos por la consulta en una caja de selección.
  if ($id_p == 1)
   {print "<option value=\"1\" selected> $nombre_p  </option>";}
  else
   {print "<option value=\"$id_p\"> $nombre_p   </option>";}
  }
 ?>

 </select> </font></td>
 </tr>
 ...
 </body>
 </html>

 <?php
 #Cierra la conexión a la base de datos
 sybase_close($conexion);
 ?>
```

Datos del programa registro2.php el cual recibirá los datos de la forma:

```
 <html><head><title>Registro</title></head>
 <body bgcolor="#FFFFFF">

 <?php
 #Archivo que contiene datos para realizar la conexión a base de datos.
 include ("./conexion.php");
 #Archivo que contiene algunas funciones de validación da datos.
 include ("./funciones.php");
 #Validación del registro de clientes
 if (ValidaNombres($nombre))
  {
  print "<center> El <b>nombre</b> que ingres&oacute; no es v&aacute;lido,
debe contener letras &uacute; nicamente.<br>";
  $error = 1;
  }
 ...
 if ($error != 1)
  {
  #Si no hay errores en los datos ingresados se procede a hacer la
inserción de datos.
  $fecha = $anio ."/". $mes ."/". $dia;
  #Primeramente se verifica que el usuario no este registrado en la tabla
AGENDA.
  #Se asigna la consulta a una variable.
  $select = "SELECT * FROM AGENDA WHERE UPPER(NOMBRE) = UPPER('$nombre')
AND UPPER(AP_PAT) = UPPER('$ap_pat') AND FECHA_NAC = '$fecha'";
  #Se manda a ejecutar la consulta.
  $query = sybase_query($select, $conexion);
```

```
  #El número de columnas devueltas por la consulta se asignan a una
variable.
  $existe = sybase_num_rows($query);
  #Si el valor de el número de columnas devuelta por la consulta es cero,
el usuario no esta registrado en la base de datos.
  if ($existe == 0)
   {
   #Se asignan los datos de la inserción a una variable.
   $insert = "INSERT INTO AGENDA (NOMBRE, AP_PAT, AP_MAT, CORREO_ELECT,
FECHA_NAC, PAIS) VALUES ('".$nombre . "','". $ap_pat . "','".$ap_mat.
"','" . $email. "','" . $fecha. "',CONVERT (NUMERIC(2), '". $pais . "'))";
   #Se ejecuta la inserción de datos.
   $query_ins = sybase_query($insert, $conexion);
   #Si fue exitosa o no la inserción se imprime en pantalla.
   if ($query_ins)
    {print "<center>El nuevo miembro qued&oacute; registrado.</center>";}
   else
    {
    print "<center>No se pudo registrar al nuevo miembro en la
agenda.</center>";
    }
   }
  else
   {
   #Si el miembro ya estaba registrado en la base de datos le informa al
usuario del sistema.
   print "<center>Esta persona ya esta registrada en la base de
datos.<br></center>";
   $error = 1;
  }
 }
 else
  {
  print "<center>Por favor presione el bot&oacute;n \"Regresar\" o
\"Back\" de su navegador y realice los cambios pertinentes.</center>";
  }
 #Cierra la conexión a la base de datos.
 sybase_close($conexion);
 ?>

 </body>
 </html>
```

JSP. Java Server Pages

Para usar aplicaciones con sevlets y jsp se deberá tener en el servidor Java Servlet, Java Server Pages y un servidor web con capacidad para Servlets como Apache Tomcat. Java Server Pages (JSP) combina HTML con fragmentos de Java para producir páginas web dinámicas. Una página JSP es un archivo de texto simple que consiste en contenido HTML o XML con elementos JSP.

Cuando un cliente pide una página JSP del sitio web y no se ha ejecutado antes, la página es inicialmente pasada al motor de JSP, el cual compila la página convirtiéndola en Servlet, la ejecuta y devuelve el contenido de los resultados al cliente. Cuando ya existe el Servlet solo se manda a llamar, sin necesidad de volver a compilarse.

Directivas

Una directiva de JSP proporciona la información del motor de JSP para la página que la pide. Su sintaxis general es <%@ directiva {atributo ="valor"} %> dónde la directiva debe tener un número de atributos.

Directivas en JSP son:

- Page: Información para la página. Posee varios atributos:
  - language="java". Comunica al servidor el lenguaje que va a ser utilizado en el archivo. Java en el único posible es esta especificación.
  - extends="package.class". La variable extends, define la clase padre del servlet generado. Normalmente no es necesario utilizar otras que no sean las clases base del proveedor.
  - import="package.*,package.class". Sirve para especificar los paquetes y clases que se quieran utilizar.
  - session="true | false". Por omisión session vale true, manteniendo los datos de las sesión para la página.
  - errorPage="pagina_error". Página que manejará las excepciones de errores.
  - isThreadSafe="true | false". Por omisión vale true, le hace señales al motor de JSP para que múltiples pedidos del cliente puedan ser tomadas como uno.
  - info="text". Información en la página a la que puede accederse a través del método Servlet.getServletInfo()
  - isErrorPage="true | false". Marca a la página como la página que manejará los errores.
  - Include: Incluye archivos completos palabra por palabra.
  - Taglib: La dirección de la biblioteca de etiquetas (tags) que se usará en la página.

Declaraciones

Una declaración de JSP, puede definirse como una definición de variables y métodos a nivel de Clase que son usadas en la página. Un bloque de declaraciones sería <%! declaración %> Un ejemplo de declaración de script sería el siguiente:

```
<HTML><HEAD><TITLE>Página JSP</TITLE></HEAD>
<BODY>
<%! String strCadena = "x"; int intContador = 0; %>
</BODY></HTML>
```

Scripts de JSP

Los scripts son bloques de código Java residentes entre las etiquetas <% y %>. Los bloques de código estarán dentro del servlet generado incluidos en método _jspService(). Los scripts pueden acceder a cualquier variable o beans que haya sido declarado. También hay algunos objetos implícitos disponibles para los Scripts desde entorno del Servlet.

Objetos

- request. Es la petición del cliente. Es normalmente una subclase de la clase HttpServletRequest.
- pageContext. Los atributos de la página y los objetos necesitan ser accesibles a través de API, para permitir al motor de JSP compilar la página. Pero cada servidor tiene implementaciones específicas de cada uno de esos atributos y objetos.
  - Para solucionar este problema, el motor de JSP utilizar la clase Factory para devolver la implementación de clase PageContext del servidor.
  - Esta clase PageContext es iniciada con los objetos response y request y algunos atributos de la directiva de la página (erropage, session, buffer y autoflush) y facilita los otros objetos para la página de petición.
- response. Es la página JSP de respuesta y es una subclase de HttpServletResponse.
- session. El objeto de sesión HTTP asociado a la petición.
- application. Lo que devuelve el servlet cuando se llama a getServletConfig().getContext()
- out. El objeto que representa la salida de texto por pantalla.
- config. El objeto ServletConfig de la página.
- page. Es la forma que tiene la página para referirse a si misma. Se usa como alternativa al objeto this.
- exception. Es una subclase libre de Throwable que es pasada a la página que maneja los errores.

El siguiente fragmento de código muestra como obtener el valor de un parámetro mediante el objeto request y como pasarlo a una cadena para mostrarlo en pantalla.

```
<%
String strNombre = request.getParameter("nombre");
out.println(strNombre);
%>
```

Expresiones de JSP

Las expresiones son una herramienta para insertar código embebido dentro de la página HTML. Cualquier cosa que este entre los tags <%= y %> será evaluado, convertido a cadena y posteriormente mostrado en pantalla. La conversión desde el tipo inicial a String es manejada automáticamente. Es importante remarcar que la expresión no termina en punto y coma (;). Esto es así porque motor de JSP, pondrá la expresión automáticamente entre out.println().

Las expresiones JSP permiten párame trizar las páginas HTML (es parecido a cuando se párame triza una consulta SQL pero difieren la forma de los valores).

Una y otra vez en el código de la página HTML, se verán ciclos o condiciones usando código Java, simplemente empezando y acabando las condiciones o ciclos entre los tags <% y %>. Un ejemplo sería:

```
<% for (int i=1;i<10;i++)
{ %>
<BR>N&uacute;mero <%=i%>
<% } %>
<BR>Termino la impresi&oacute;n de n&uacute;meros.
```

Estructuras de control en JSP

IF

Usa la misma lógica que otros lenguajes. Si a fuese mayor que b se mostraría “a es mayor que b” en pantalla:

```
if (a > b)
{ out.println("a es mayor que b");}
```

else funciona de la misma manera que en PHP.

ELSE IF

Es una extención del if, funciona como el elseif de PHP pero a diferencia de este ambas palabras van separadas. Ejemplo:

```
if (a > b)
{ printf ("a es mayor que b");}
else if (a == b)
{ printf ("a es igual que b");}
```

WHILE

Funciona de la misma forma que en otros lenguajes. Ejemplo:

```
a=1;
b=3;
while (a < b)
{
out.println("a es menor que b");
a++;
}
```

DO ... WHILE

Esta sentencia siempre se ejecutará al menos una vez. Ejemplo:

```
a=5;
b=3;
do
{ out.println ("a es mayor que b"); a--; }
while (a > b);
```

FOR

Su sintaxis es:

```
for (expr1; expr2; expr3)
for (int i = 1; i < 10; i++)
{ out.println (i + " "); }
```

SWITCH

```
switch (expr)
{
case valor1:
sentencias;
break;
default:
sentencias;
break;
}

switch (i)
```

```
{
case 0:
out.println ("i es igual a 0");
break;
case 1:
out.println ("i es igual a 1");
break;
case 2:
out.println ("i es igual a 2");
break;
}
```

BREAK
Escapa de las estructuras de control cíclicas o del switch.

Para trabajar con la base de datos

Para trabajar con la base de datos es necesario realizar estos siete pasos:

1. Cargar el controlador JDBC.

El controlador es la pieza de software que sabe como hablar con la base de datos. Para cargar el controlador todo lo que se tiene que hacer es cargar la clase apropiada. Usamos el método Class.forName para pasarle la cadena que representa al nombre de la clase correcta para hacer la conexión, el método carga la clase correspondiente.

Esta línea cargaría el controlador para un manejador Sybase.

```
Class.forName("com.sybase.jdbc.SybDriver");
```

Se puede usar este método para cualquier clase en el CLASSPATH. La mayoría de los vendedores distribuye sus controladores dentro de archivos JAR, debemos estar seguros de que el archivo JAR este definido en el CLASSPATH.

2. Definir la conexión URL.

Después de cargar el controlador JDBC es necesario especificar el lugar en donde se encuentra el servidor de la base de datos. Por URL nos referimos al protocolo jdbc: y al nombre del servidor, puerto y nombre de la base de datos.

El formato exacto de un URL esta definido en la documentación de cada controlador, a continuación se encuentran ejemplos:

```
  String nombreServ = "servidor.basededatos.com";
  String nombreBd = "miBase";
  int puerto = 1234;
  String oracleURL = "jdbc:oracle:thin:@" + nombreServ + ":" + puerto +
":" + nombreBd;
  String sybaseURL = "jdbc:sybase:Tds:" + nombreServ + ":" + puerto + "/"
+ nombreBd;
```

3. Establecer la conexión.

Es necesario pasar al método getConnection de la clase DriverManager el URL, el usuario de la base de datos y la contraseña.

```
  String usuario = "miUsusario";
  String pwd = "miContraseña";
  Connection conexion = DriverManager.getConnection (sybaseURL, usuario,
pwd);
```

La clase Connection incluye los métodos:

- Ø prepareStatement. crea una consulta predefinida,
- Ø rollback. deshace las consultas a partir del último commit.
- Ø Commit. termina las consultas desde el último commit,
- Ø isClosed. revisa si la conexión fue cerrada.

4. Crear el objeto statement.

El objeto Statement es usado para enviar las consultas y comandos a la base de datos y es creado a partir de Connection:

```
Statement statement = conexion.createStatement();
```

5. Ejecutar la sentencia SQL.

Una vez creado el objeto Statement, se puede usar para enviar consultas SQL utilizando el método executeQuery, el cual regresa un objeto de tipo ResultSet.

```
  String consulta = "SELECT columna1 FROM miTabla";
  ResultSet rs = statement.executeQuery(consulta);
```

Para modificar la base de datos (UPDATE, INSERT o DELETE) se usa el método executeUpdate.

Además se pueden crear consultas parametrizadas, si es que vamos a ejecutar código SQL similar varias veces usando los statements precompilados (prepared statements). La idea es crear un objeto parametrizado en una forma estándar que es enviado a la base de datos para hacer la compilación antes de ser usada. Se utiliza un signo de interrogación para indicar los lugares en donde el valor va a ser sustituido en el statement.

Cada vez que se utilice el statement precompilado simplemente se remplazan los valores con una llamada a setXXXX que corresponda al tipo de parámetro que se desea usar (setInt, setString) y a la entrada del campo. Ejemplo:

```
  Connection conexion = DriverManager.getConnection (sybaseURL,usuario,
pwd); String template = "UPDATE  miTabla SET colFlotante = ? "+" WHERE
colEntero = ? ";
PreparedStatement actualiza = conexion.prepareStatement (template);
  actualiza.setDouble(45.67); actualiza.setInt(4);
  actualiza.executeUpdate();
```

6. Procesar los resultados.

La manera más simple de obtener los resultados es procesarlos uno a la vez usando el método next del ResultSet, que se mueve uno a uno por los renglones devueltos por la consulta.

ResultSet proporciona varios métodos getXXXX que toman como argumento el índice o el nombre de una columna.

```
  while(rs.next()){
  System.out.println(rs.getInt(1) + " " + rs.getString(2));
}
```

7. Cerrar la conexión.

Solo hay que poner la siguiente línea:

```
  conexion.close();
```

Ejemplo más completo de un programa:

```
 <%@ page contentType="text/html;charset=windows-1252"
 import ="java.sql.*"
 %>
 <%
 try { Class.forName("com.sybase.jdbc2.jdbc.SybDriver"); }
 catch (ClassNotFoundException e)
 {out.println ("<h1>No se encontr&oacute; el controlador:" + e +
e.getMessage() + "</h1>" ); }
 try {
  String strSQL;
  String host = "dirección IP";
  String port = "puerto de Sybase";
  String url = "jdbc:sybase:Tds:" + host + ":" + port;
  Connection conexion = DriverManager.getConnection
(url,"usuario","password");
  Statement Stm = conexion.createStatement();
  strSQL = "CREATE TABLE AGENDA (ID NUMERIC(3) IDENTITY PRIMARY KEY,
NOMBRE VARCHAR(30) NOT NULL, AP_PAT VARCHAR(25) NOT NULL, AP_MAT
VARCHAR(25) NULL, CORREO _ELECT VARCHAR(90) NOT NULL, FECHA_NAC DATETIME,
PAIS NUMERIC(2) NOT NULL)";
  Stm.executeUpdate(strSQL);
  conexion.close();
  } catch (Exception e) {
 out.println( "<h1>Excepci&oacute;n:"+e+e.getMessage()+"</h1>" ); }
 %>
```

Lo más común es mostrar los resultados en pantalla de una sentencia SELECT, lo podemos hacer de la siguiente forma:

```
 <%@ page contentType="text/html;charset=windows-1252"
 import ="java.sql.*"
 %>
 <%
 try {Class.forName("com.sybase.jdbc2.jdbc.SybDriver");
  } catch (ClassNotFoundException e) {
  out.println("<h1>No se encontr&oacute; el controlador:" + e +
e.getMessage() + "</h1>" );
          }
 try {String host = "dirección IP";
  String port = "puerto de Sybase";
  String url = "jdbc:sybase:Tds:" + host + ":" + port;
```

```
  Connection conexion = DriverManager.getConnection
(url,"usuario","password");
  Statement st = conexion.createStatement();
  String select = "SELECT ID, NOMBRE, AP_PAT, AP_MAT FROM AGENDA ORDER BY
AP_PAT";
  ResultSet rs = st.executeQuery(select);
  out.println("<TABLE>");
  out.println("<TR>");
  while (rs.next())
  {
 out.println("<TR>");
 out.println("<TD>" + rs.getInt(1) + "</TD>");
 out.println("<TD>" + rs.getString(2) + "</TD>");
 out.println("<TD>" + rs.getString(3) + "</TD>");
 out.println("<TD>" + rs.getString(4) + "</TD>");
 out.println("</TR>");
  }
 out.println("</TABLE>");
 conexion.close();
 } catch (Exception e) {
 out.println( "<h1>Excepci&oacute;n: "+e+e.getMessage()+"</h1>" ); }
 %>
```

ASP. Active Server Pages

ASP es una tecnología creada por Microsoft con la que el usuario puede recibir páginas generadas dinámicamente en el servidor. Estas páginas contienen código HTML y fragmentos de código que son utilizados para llevar a cabo la interacción entre el servidor y el navegador.

ASP puede conectarse a cualquier motor que disponga de ODBC. Para procesar una página ASP es necesario un servidor Web de Microsoft. Se utiliza el archivo ASP.DLL para interpretar el código, siendo el servidor más extendido Internet Information Server (IIS).

Para plataformas Unix es necesario añadir un software que actúe de intérprete siendo algunos de los más conocidos: Chilisoft y Instant ASP

El código ASP va entre las etiquetas <% y %>, esto le indicará al servidor cual es la parte que debe procesar.

Contenido de una página ASP

Se puede elegir de entre dos lenguajes con cual trabajar los programas ASP, estos lenguajes son VBScript y Jscript el más usado es el primero y tiene su origen en el lenguaje de programación Visual Basic, Jscript se parece más a JavaScript.

En la propiedad LANGUAGE deberá especificarse el tipo de lenguaje a utilizar.
`<%@ LANGUAGE="VBSCRIPT" %> o <%@ LANGUAGE="JSCRIPT" %>`

Para especificar un comentario en una página ASP se debe introducir una comilla simple o la palabra REM.

```
<%
REM Así se documentan las secciones de los programas
'Cualquier comentario que no, será interpretado por el servidor
%>
```

El tipo de dato que utiliza VBSCRIPT es Variant que es una clase especial de datos que puede contener diferentes tipos de información, se comporta como un número cuando se utiliza en un contexto numérico y como una cadena de caracteres cuando se usa en un contexto de cadena, no obstante podemos forzar a que los números se comporten como cadenas poniéndolos entre comillas (" "). Las cadenas se asignan entre comillas: Variable = "Hola", Los números van sin comillas: Variable = 22 y Las fechas van entre el símbolo de número: Variable = #12-1-2003#

La declaración de variables es opcional, aunque su práctica es una buena costumbre ya que se evitan errores y se facilita la lectura del código, se puede forzar la declaración incluyendo la sentencia <% Option Explicit %>.

En la declaración se utiliza la palabra reservada Dim se pueden anidar varias declaraciones mediante el separador ",".

```
<% Dim Variable %>
```

La declaración de arreglos es igual, la única diferencia es que éstos llevan paréntesis a continuación del nombre de la variable, dentro de los paréntesis deberá ir el número de elementos que habrá en el arreglo.

```
Dim Mi_Arreglo(5)
```

Los elementos del arreglo serán enumerados a partir de cero. Así Mi_Arreglo(5) tendría seis elementos del cero al cinco. Los arreglos pueden cambiar de dimensión (arreglos dinámicos), el arreglo debe declararse sin número de dimensiones:

```
Dim ArregloDin()
```

Con la sentencia Redim se determinan las dimensiones:

```
Redim ArregloDin(20)
```

Si deseamos conservar los valores almacenados en el arreglo cuando realizamos el redimensionamiento, se deberá añadir la sentencia Preserve.

```
Preserve:
Redim ArregloDin(20)
.......
Redim Preserve ArregloDin(26)
```

Las constantes se definen con la sentencia CONST Const *NombreConstante* = *Valor_que_no_cambia*

Estructuras de control ASP

IF THEN ELSE

Su sintaxis es:

```
 If expr Then
  sentencias
 Else
  sentencias
 End If
```

Puede o no ocuparse la estructura Else. Puede ponerse else if ó elseif.

WHILE ... WEND

Funciona como los otros lenguajes. Su sintaxis es:

```
  While expr
  sentencias
  Wend
```

FOR

Realiza ciertas instrucciones una determinada cantidad de veces. Su sintaxis es:

```
  For inicio to fin
  sentencias
  Next
```

Objetos

Los objetos son programas compilados e instalados en el servidor y que han sido programados para realizar un conjunto de operaciones fácilmente accesibles por otros programas.

Objeto Application: Se utiliza para compartir información entre todos los usuarios de una aplicación. Tiene dos métodos

- Lock. Ayuda a asegurarnos que otros clientes no puedan modificar ninguna de las variables y propiedades de la aplicación hasta que no se llame el método
- Unlock. Anula el bloqueo del objeto Application para el resto de clientes.

Objeto Request: El objeto Request se utiliza para tener acceso a la información que se pasa en las peticiones HTTP. Entre dicha información se incluyen los parámetros que se pasan desde los formularios HTML mediante el método POST o el método GET, cookies y certificados de cliente. Los métodos más usados son:

- Ø FORM. Al consultar formularios los pares nombre/valor se guardan aquí.
- Ø QUERYSTRING. En él, la información en forma de pares nombre/valor se agrega y transmite a través de la URL, justo después de la dirección de una página HTML, se guarda ahí y si es preciso se proporciona.
- Ø COOKIES. Aquí se guarda información referente a las cookies, si el cliente remite cookies.
- Ø SERVERVARIABLES. Conserva en listas los valores de variables del servidor HTTP.
- Ø CLIENTCERTIFICATE. Lista de valores utilizada para certificados y datos de seguridad.

Objeto Response: El objeto Response se utiliza para controlar la información enviada al usuario. Esto incluye el envío de información directamente al explorador, la redirección del explorador a otra dirección URL o el establecimiento de valores de las cookies. Algunos de sus métodos son:

- Ø WRITE. Envía información al navegador.
- Ø REDIRECT. Con él se pueden elegir otras páginas ASP pero también páginas HTML sencillas.

Objeto Server: El objeto Server proporciona acceso a los métodos y las propiedades del servidor. El método utilizado más frecuentemente es el que crea una instancia de un componente ActiveX (Server.CreateObject). Con la propiedad Timeout establecemos cuanto tiempo puede tardarse en ejecutar una página.

Objeto Session: El objeto Session permite almacenar la información necesaria para una determinada sesión de usuario. Las variables almacenadas en el objeto Session no se descartan cuando el usuario pasa de una página a otra dentro de la aplicación, si no que dichas variables persisten durante todo el tiempo que el usuario tiene acceso a las páginas de la aplicación. También puede utilizar los métodos de Session para terminar explícitamente una sesión y establecer el periodo de tiempo de espera de inactividad de las sesiones.

DSN

Conexión con DSN

1.- La conexión con DSN es la más cómoda, pero sólo se puede utilizar si se tiene acceso al Panel de Control de la máquina servidor.
2.- Creamos nuestra base de Datos en Access y la guardamos.
3.- Luego vamos a Inicio > Configuración > Panel de Control y allí elegimos Fuentes de Datos ODBC.
4.- Al ingresar nos encontramos con una pantalla que es el administrador de orígenes de datos ODBC. En la solapa DSN de Usuario presionamos el botón Agregar.
5.- Luego seleccionamos Microsoft Access Driver (*.mdb) y presionamos Finalizar. Ahora se hará la conexión ODBC.
6.- Presionamos el botón Seleccionar y elegimos nuestra Base de Datos e ingresamos el nombre de la base en el primer campo.
7.- Por último el botón Aceptar.

EJEMPLO:

```
<%
'Definimos la variable para la conexión.
Dim Conex
Set Conex = Server.CreateObject ("ADODB.Connection")
'Y ya estamos conectados a nuestra base de datos.
Conex.Open "nombre de la BD"
'Se continúa con el programa.
%>
```

Conexión sin DSN

Este tipo de conexión es más “complicada”, se hace la conexión a la base de datos mediante comandos.:

```
<%
Dim Conex
'Creamos el objeto de conexión
Set Conex = Server.CreateObject ("ADODB.Connection")
Conex.Open "driver={SQL Server};server=TU_SERVIDOR;
database=NOMBRE_BASE;uid=sa;pwd=xx"
%>
```

Para mostrar datos de la base:

```
<%
 Dim Conexion, RS
 'Abrir la conexión a la base de datos.
 Set Conexion = Server.CreateObject ("ADODB.Connection")
 Conexion.Open ("DBQ=" & server.mappath ("base_datos.mdb") &
";Driver={Microsoft Access Driver (*.mdb)};")
 'Ejecutar consultas en la BD.
 Set RS = Server.CreateObject ("ADODB.RecordSet")
  'Entre comillas va el nombre de la tabla y luego de la coma se indica la
BD que contiene esa tabla.
  ' El número que le sigue, el "1", es la forma de abrir la tabla 1 es solo
lectura.
 RS.Open "persona", Conexion, 1
 'Mostramos los campos mientras que el RS no termine.
 Do While Not RS.EOF
 Response.Write RS("id") &" "
 Response.Write RS("nombre") &" "
 Response.Write RS("ap_pat")&" "
 Response.Write RS("ap_mat") & "<BR>"
'MoveNext nos moverá al siguiente registro de la tabla.
 RS.MoveNext
 Loop 'primero cerramos los objetos y luego los limpiamos.
 RS.Close
 Conexion.Close
 Set RS = nothing
 Set Conexion = nothing
 %>
```

El objeto Recordset

Tipos de cursores

El cursor es el puntero que nos va a permitir desplazarnos por los registros del recordset. Dependiendo del tipo de cursor será la manera en que podremos movernos por los datos o hacer cambios a los mismos.

Valor	Características
0	Solo nos permite recorrer la tabla en forma secuencial. No podemos movernos hacia atrás y no permite modificaciones.
1	Permite modificaciones y permite movernos en ambos sentidos. Se piden ver los cambios realizados por otro recordset a excepción de las altas.
2	Igual al anterior solo que aquí si vemos todos los cambios realizados por otro recordset.
3	Nos permite movernos en dos sentidos y no permite modificaciones. No se ven los cambios a la tabla originados por otro recordset.

Tipos de bloqueo

Esto nos sirve para evitar que dos usuarios modifiquen el mismo registro a la vez.

Valor	Características
1	No permite modificar los datos de la tabla.
2	Cuando se abra la tabla nadie màs podrá hacerlo.
3	Cierra la tabla cuando se invoque al método Update del objeto Recorset.

‘Abre el recordset para que se puedan hacer modificaciones y para que cuando se abra la tabla nadie más la abra.

```
  rs.Open sql,Conexion, 1, 2
 'Asigna valor de las variables a los campos de la tabla proveedores
  rs.Fields("nombre").value = nombre
```

'Actualiza el registro del recordset tras haberlo modificado rs.update

Capítulo V. Mejores Prácticas en la Administración de Sistemas

Administración de Sistemas

Conceptos de Auditoría Informática

Concepto de Auditoría 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 Auditoría 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 empresa

Conforme han avanzando las teorías de administración de empresas, el papel y la percepción de la auditoría en las organizaciones fue cambiando, aunque lamentablemente, no con la velocidad necesaria.

Hoy día, debemos 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 empresa.

La auditoría 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 Auditoría Informática

- Ø 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.
- Ø 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.
- Ø 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.
- Ø 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.

- Ø 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.
- Ø 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 en la tecnología deben adecuarse a esta situación evitando utilizar recursos en tecnología que no aportan de manera sustancial a la organización o descuidar inversiones por una austeridad extrema no justificada. En resumen, las inversiones deben responder a un análisis de costo beneficio para la organización.

Proceso de Auditoría

La metodología que se emplea en la Auditoría Informática es similar a las fases que componen una auditoría tradicional:

Roles y Responsabilidades

Las estructuras organizacionales (organigramas) en las áreas de TI difieren de organización a organización; sin embargo, en cualquiera de ellas son un elemento importante para que todos los empleados tengan un conocimiento claro de la responsabilidad, autoridad, roles y responsabilidades.

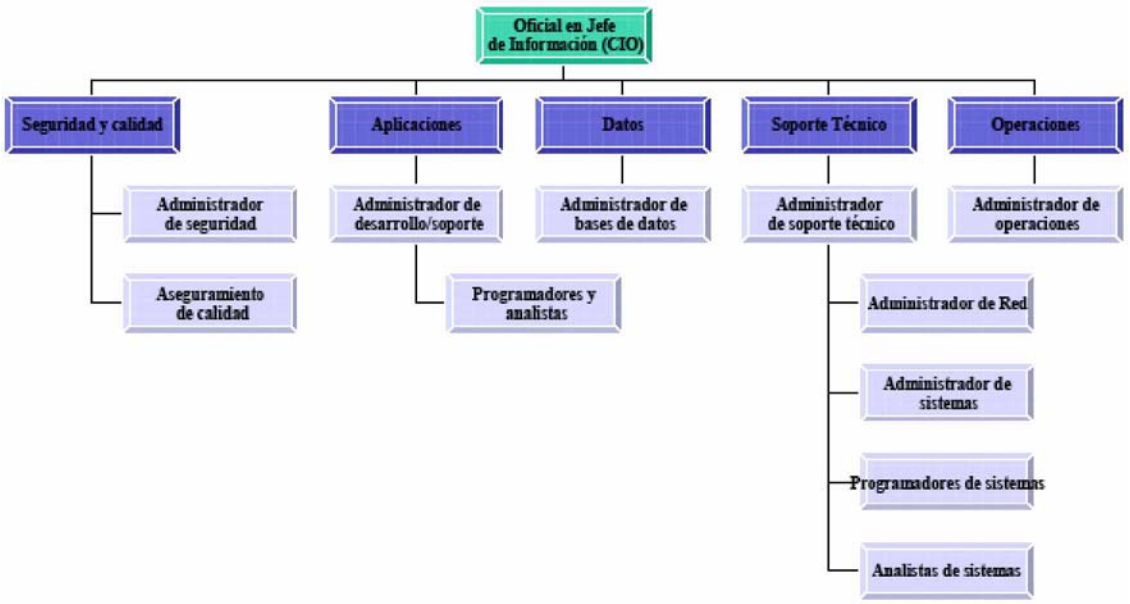
Funciones del administrador de bases de datos

De acuerdo al manual de revisión de los Certified Information Systems Auditor (CISA) las responsabilidades del administrador de bases de datos son:

- Ø Custodia información de la organización.
- Ø Debe comprender a la empresa, datos de usuario y las relaciones de éstos.
- Ø Responsable de la seguridad y clasificación de la información de los datos compartidos almacenados en los sistemas de BD.
- Ø Responsable del diseño real, definición y mantenimiento de las BD corporativas.
- Ø Especificar la definición física de los datos y cambiarla para su mejor desempeño.
- Ø Seleccionar e implementar herramientas de optimización de la BD.
- Ø Probar y evaluar las herramientas de programadores.
- Ø Dar soporte técnico a programadores sobre estructura de la BD.
- Ø Implementar controles de definición, acceso, actualización y concurrencia.
- Ø Monitorear el uso, recopilar estadísticas de desempeño y ajustar la BD.
- Ø Definir e iniciar los procedimientos de respaldo y recuperación.

Segregación de funciones

Organización de un departamento de sistemas de Información (puestos y organigrama pueden variar entre empresas)



En la segregación de funciones el auditor deberá determinar la relación entre las funciones, responsabilidad y autoridad.

Una adecuada segregación de funciones

- Ø Evita que una sola persona pueda ser responsable de funciones diversas y críticas.
- Ø Evita que se cometan errores o apropiaciones indebidas difíciles de detectar.
- Ø Previene y disuade actos fraudulentos o maliciosos.
- Ø Puede restringir acceso a: la computadora, Biblioteca de datos de producción, Programas de producción, Documentación de programas, Sistema Operativo y utilerías.
- Ø Reduce el daño potencial por acciones de personas.

En empresas pequeñas, debe haber controles compensatorios, para mitigar el riesgo de no tener esta segregación. Control Compensatorio es un control interno que reduce el riesgo de una debilidad de control.

	Gpo Cnt	Ana Sis	Prog Apl	Help Desk	Usua Fin	Ing Dat	Oper Com	Adm BD	Adm Red	Adm Sis	Adm Seg	Cint	Prog Sis	Cnt Cal
Gpo Cnt		X	X	X		X	X	X	X	X			X	
Ana Sis	X			X	X		X				X	X		
Prog Apl	X			X	X	X	X	X	X	X	X	X	X	
Help Desk	X	X	X		X	X		X	X	X		X	X	
Usua Fin		X	X	X			X	X	X			X	X	X
Ing Dat	X		X	X			X	X	X	X	X		X	
Oper Com	X	X	X		X	X		X	X	X	X		X	
Adm BD	X		X	X	X	X	X		X	X			X	
Adm Red	X		X	X	X	X	X	X				X		
Adm Sis	X		X	X		X	X	X				X		
Adm Seg		X	X			X	X					X	X	
Cint		X	X	X	X				X	X	X		X	
Prog Sis	X		X	X	X	X	X	X			X	X		X
Cnt Cal					X								X	

X= Combinación de estas funciones pueden crear una debilidad potencial de control
Esta matriz de control es solo una guía

Controles para el reforzamiento de la segregación de funciones

Pueden usarse diversos mecanismos para el reforzamiento de la segregación de funciones:

Acceso a datos

Controles sobre acceso a datos son provistos por una combinación de seguridad física, de sistema y de aplicación en el área del usuario y en el centro de procesamiento de datos.

El ambiente físico debe ser asegurado para prevenir el acceso no autorizado de personal a los diversos dispositivos tangibles conectados a la unidad central de procesamiento y por lo tanto, prevenir el acceso a los datos.

La seguridad en los sistemas y aplicaciones son capas adicionales de seguridad que pueden prever que individuos no autorizados obtengan acceso a datos de la organización.

Acceso a datos desde conexiones externas es una preocupación creciente desde el advenimiento de la Internet.

Las decisiones de control de acceso están basadas en políticas organizacionales y en dos estándares aceptados en la práctica: la separación de funciones y el menor privilegio.

Los controles para que se usen efectivamente no deben interrumpir más de lo necesario el flujo usual de trabajo o establecer demasiada carga a los administradores, auditores y usuarios autorizados. Aún más, el acceso no debe ser incondicional y los controles de acceso deben proteger adecuadamente todos los recursos organizacionales; para asegurar esto es necesario primero categorizar los recursos.

Las políticas establecen niveles de sensibilidad para los datos y para los recursos tales como: ultra secreto, secreto, confidencial y no clasificado. Estos niveles deberían ser usados como guía en los procedimientos adecuados para el manejo de los activos de información. Esto también puede ser tomado en cuenta como base para el control de acceso. A los individuos sólo se les garantiza el acceso a recursos específicos o al nivel más bajo de sensibilidad.

Las etiquetas se usan para indicar el nivel de sensibilidad de documentos almacenados electrónicamente. Los controles basados en políticas pueden ser mandatarios o discrecionales.

Autorización de transacciones. Es responsabilidad de los usuarios. La autorización es delegada al grado que sea relativo al nivel particular de responsabilidad del individuo autorizado en el departamento de usuario. Verificaciones periódicas deben ser desempeñadas por la administración y por auditoría para detectar la entrada autorizada de transacciones.

Custodia de activos corporativos. Estos deben ser determinados y asignados apropiadamente. El dueño de los datos es asignado a un usuario de departamento en particular y sus obligaciones deberían ser especificadas por escrito. El propietario de los datos tiene la responsabilidad para determinar los niveles de autorización requeridos para proveer seguridad adecuada, mientras que el grupo de administración es frecuentemente responsable de la implementación y reforzamiento del sistema de seguridad.

Formas de autorización

Los administradores de los departamentos usuarios deben proveer a Sistema de Información de formas de autorización formal que definan los derechos de acceso a cada uno de sus colaboradores; en otras palabras, los responsables de los usuarios deben definir quién tendría acceso a qué.

Las formas de autorización deben ser evidenciadas adecuadamente con el nivel de aprobación de la administración. De manera general, todos los usuarios deberían estar autorizados con acceso específico a los sistemas vía solicitud formal de la administración.

En compañías muy grandes o en aquellas con sitios remotos, deberían mantenerse catálogos de firmas de autorización y las solicitudes formales deberían ser comparadas con dicho catálogo.

Los derechos de acceso (privilegios) deben ser revisados periódicamente para asegurar que están actualizados y son adecuados a las funciones del trabajo del empleado.

Tablas o registros de autorización de usuarios

El departamento de Sistemas de Información debería usar los datos de las formas de autorización para construir y mantener tablas de autorización de usuarios. Estas tablas indicarán quiénes están autorizados para actualizar, modificar, borrar y/o ver datos. Estos privilegios se proveen a nivel sistema, transacción o campo; en efecto, estas son las listas de control de acceso.

Las tablas de autorización deben ser aseguradas contra acceso no autorizado mediante protección adicional con contraseñas o encriptación de datos.

La bitácora de control debe registrar toda la actividad del usuario y el nivel apropiado de la administración debería revisarla periódicamente; todas las excepciones deben ser investigadas.

Controles compensatorios por falta de segregación de funciones. En organizaciones muy pequeñas en donde el departamento de Sistemas de Información consiste en cuatro o cinco personas deben existir medidas de control compensatorio para mitigar el riesgo resultante de la falta de la segregación de funciones. Algunos de los controles compensatorios son:

- Pistas (Registros) de auditoría. Son un componente esencial en todos los sistemas bien diseñados. Éstas ayudan al departamento de Sistemas de Información al proveer un mapa para rastrear el flujo de una transacción. Esto permite al usuario y al auditor a recrear el flujo de transacción actual desde su punto de origen hasta su existencia o la actualización de un archivo. En la ausencia de una adecuada segregación de funciones, los registros de auditoría pueden ser aceptadas como un control compensatorio aceptable. El revisor debería ser capaz de determinar quien inicio la transacción, la hora del día y la fecha de ingreso, el tipo de ingreso, qué campos de información contenía y qué archivos actualizaba.
- Conciliación. Es responsabilidad última del usuario. En algunas organizaciones, la conciliación limitada de aplicaciones puede ser ejecutada por el grupo de control de datos con el uso de totales de control y hojas de balance. Este tipo de verificación independiente incrementa el nivel de confianza que la aplicación corrió adecuadamente y que los datos fueron adecuadamente balanceados.
- Reportes de excepción. Deben ser manejados a nivel supervisión y deberían requerir evidencias, tales como iniciales en un reporte subrayando que la excepción fue tratada adecuadamente. La administración debería asegurarse que las excepciones son resueltas de forma oportuna.
- Bitácoras de transacción. Pueden ser manuales o automáticas. Un ejemplo de bitácora manual es el registro de transacciones en batch antes de que sean enviadas a procesamiento. Una bitácora de transacción automática provee un registro de todas las transacciones procesadas y es mantenida por el sistema operativo.
- Revisiones de supervisión. Pueden ser ejecutadas a través de la observación y preguntas o remotamente.
- Revisiones independientes. Se llevan a cabo para compensar los errores o fallas intencionales al seguir los procedimientos prescritos. Estas son particularmente importantes cuando las obligaciones en organizaciones muy pequeñas no pueden ser apropiadamente segregadas. Dichas revisiones ayudarán a detectar errores o irregularidades.

ISACA, Objetivos de Control, COBIT

ISACA

¿Cómo podrían asegurar las organizaciones, que construyen proyectos de tecnología de información, que están cumpliendo adecuadamente con las necesidades del cliente, en forma eficiente y oportuna y dentro del presupuesto contemplado?

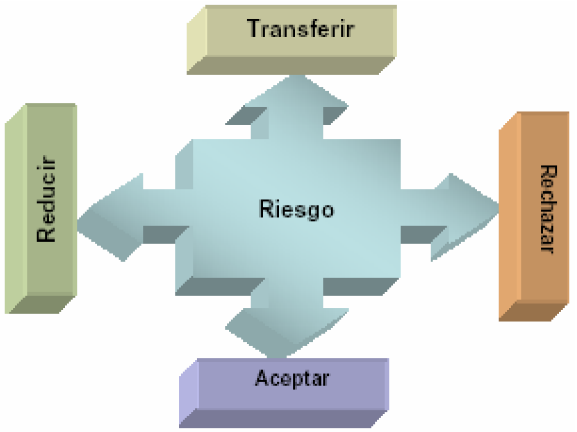
Existe una asociación internacional denominada Information Systems Audit and Control Association (ISACA); comenzó en 1967 con un grupo pequeño de profesionales con actividades similares; verificar controles en los sistemas de cómputo que se estaban convirtiendo en parte crítica en las operaciones de 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 auditoría 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.

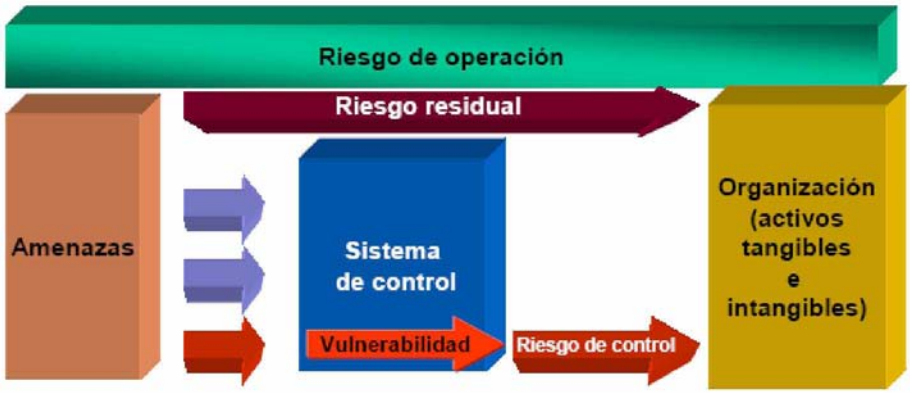
Objetivos de control

Los activos de información (información, aplicaciones, infraestructura y gente) están sometidos a una serie de amenazas del entorno. Dichas amenazas son valoradas por la administración de la organización para establecer salvaguardas o controles.

Las debilidades en los controles son conocidas como vulnerabilidades. De estos conceptos se desprenden el análisis y la administración de riesgos. De acuerdo con las guías para la administración de la seguridad de TI un riesgo es “El potencial de que una amenaza dada explote vulnerabilidades de un activo o grupos de activos para causar pérdidas o daños a tales activos. El impacto o la severidad relativa del riesgo es proporcional a la pérdida o daño al valor del negocio y a la frecuencia estimada de la amenaza”.



El control es una medida para reducir los riesgos; se materializan en las políticas, procedimientos, prácticas y estructuras organizacionales establecidas por la organización para lograr sus objetivos y para que los eventos no deseados serán prevenidos o detectados y corregidos.



* Errores	* Falta de conocimiento	* Impactos
* Ataques / Daño malicioso	* Falta de Seguridad funcional	* Pérdida de dinero
* Fraude	* Tecnología no probada	* Incumplimiento normativo
* Robo		* Perdida de Reputación
* Falla de infraestructura	* Comunicación sin protección	* Violación de confianza
		* Reducción en la eficiencia
		* Interrupciones en la operación

La clasificación de los controles según su naturaleza es:

Clase	Función	Ejemplos
Preventivo	Ø Detectar problemas antes de que surjan. Ø Monitorear las operaciones y las entradas. Ø Intentar predecir problemas potenciales antes de que ocurran y se hagan ajustes. Ø Prevenir que ocurran errores, omisiones o actos maliciosos	Ø Emplear sólo personal calificado. Ø Segregar funciones. Ø Controlar el acceso a instalaciones físicas. Ø Usar documentos bien diseñados. Ø Establecer procedimientos para autorizar transacciones
Detectivo	Ø Usar controles que detecten y reporten la ocurrencia de un error, omisión o acto malicioso	Ø Totales hash Ø Puntos de verificación en los Jobs de producción Ø Verificación duplicada de cálculos Ø Funciones de auditoría interna
Correctivo	Ø Minimizar el impacto de una amenaza Ø Remediar los problemas	Ø Planes de contingencia Ø Procedimientos de respaldo Ø Procedimientos de recorrida

	descubiertos por controles detectivos Ø Identificar las causas de los problemas Ø Corregir errores que surjan de los problemas Ø Modificar los sistemas de procesamiento para minimizar las ocurrencias del problema en el futuro	

Los objetivos de control (procesos de control) son materializados por los controles específicos. En otras palabras un objetivo de control es una declaración de un propósito o resultado deseable a ser alcanzado mediante la implementación de prácticas de control en una actividad de TIC en particular.

Control Objectives for Information and Related Technology (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:

Ø Representan el consenso de expertos fuertemente enfocados en el control y menos en la ejecución.
Ø 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.

Ø Planeación y Organización
Ø Adquisición e Implementación
Ø Entrega y Soporte
Ø Monitoreo y Evaluación

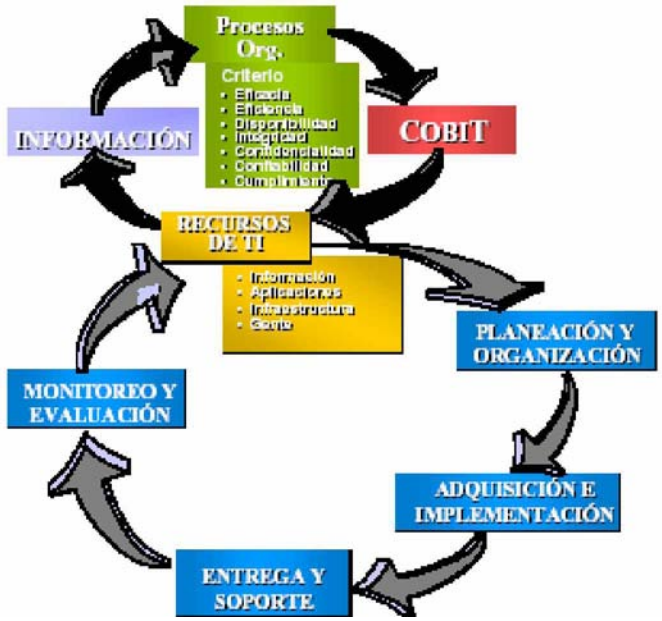
Considera las necesidades fiduciarias, de calidad y de seguridad de una organización proviendo 7 criterios de información que pueden ser usados para genéricamente definir qué requiere de TI la organización.

Ø Eficacia
Ø Eficiencia
Ø Disponibilidad
Ø Integridad
Ø Confidencialidad
Ø Confiabilidad
Ø Cumplimiento

Características de COBIT

COBIT cubre cuatro características básicas:

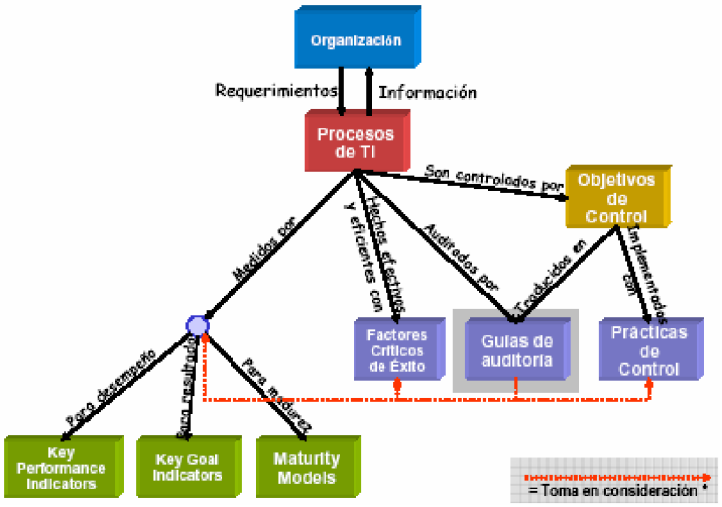
Para proveer la información que la organización necesita para lograr sus objetivos, los recursos de TI necesitan ser administrados por una serie de procesos naturalmente agrupados.



EVALUACIÓN

La estructura documental de COBIT respecto de las áreas y niveles de organización puede verse representada en la siguiente imagen:

La forma de operar de COBIT puede verse representada en la siguiente imagen:



Planeación y Organización

PO1 Definir un plan estratégico para TI. Se requiere una planeación estratégica de TI para administrar y dirigir todos los recursos de TI de acuerdo con la estrategia del negocio y las prioridades. La función de TI y los participantes del negocio son responsables de garantizar que se materialice el valor óptimo de los portafolios de proyectos y servicios. El plan estratégico debe mejorar el entendimiento de los interesados clave respecto a las oportunidades y limitaciones de TI, evaluar el desempeño actual y aclarar el nivel de inversión requerido. La estrategia de negocio y las prioridades se deben reflejar en los portafolios y deben ser ejecutadas por los planes tácticos de TI, los cuales establecen objetivos, planes y tareas específicas, entendidas y aceptadas tanto por el negocio como por TI.

PO2 Definir la Arquitectura de la Información. La función de los sistemas de información debe crear y actualizar de forma regular un modelo de información del negocio y definir los sistemas apropiados para optimizar el uso de esta información. Esto incluye el desarrollo de un diccionario corporativo de datos que contiene las reglas de sintaxis de los datos de la organización, el esquema de clasificación de datos y los niveles de seguridad. Este proceso mejora la calidad de la toma de decisiones gerenciales asegurándose que se proporciona información confiable y segura y permite racionalizar los recursos de los sistemas de información para igualarse con las estrategias del negocio. Este proceso de TI también es necesario para incrementar la responsabilidad sobre la integridad y seguridad de los datos y para mejorar la efectividad y control de la información compartida a lo largo de las aplicaciones y de las entidades

PO3 Determinar la dirección tecnológica. La función de servicios de información debe determinar la dirección tecnológica para dar soporte al negocio. Esto requiere de la creación de un plan de infraestructura tecnológica y de un consejo de arquitectura que establezca y administre expectativas realistas y claras de lo que la tecnología puede ofrecer en términos de productos, servicios y mecanismos de aplicación. El plan se debe actualizar de forma regular y abarca aspectos tales como arquitectura de sistemas, dirección tecnológica, planes de adquisición, estándares, estrategias de migración y contingencias. Esto permite contar con respuestas oportunas a cambios en el ambiente competitivo, economías de escala para consecución de personal de sistemas de información e inversiones, así como una interoperabilidad mejorada de las plataformas y de las aplicaciones.

PO4 Definir los procesos, organización y relaciones de TI. Una organización de TI se debe definir tomando en cuenta los requerimientos de personal, funciones, delegación, autoridad, roles, responsabilidades y supervisión. La organización estará incrustada en un marco de trabajo de procesos de TI que asegura la transparencia y el control, así como el involucramiento de los altos ejecutivos y de la gerencia del negocio. Un comité estratégico debe garantizar la vigilancia del consejo directivo sobre la TI y uno ó más comités administrativos, en los cuales participan tanto el negocio como TI, deben determinar las prioridades de los recursos de TI alineados con las necesidades del negocio. Deben existir procesos, políticas administrativas y procedimientos para todas las funciones, con atención específica en el control, el aseguramiento de la calidad, la administración de riesgos, la seguridad de la información, la propiedad de datos y de sistemas y la segregación de tareas. Para garantizar el soporte oportuno de los requerimientos del negocio, TI se debe involucrar en los procesos importantes de decisión.

PO5 Administrar la inversión en TI. Establecer y mantener un marco de trabajo para administrar los programas de inversión en TI que abarquen costos, beneficios, prioridades dentro del presupuesto, un proceso presupuestal formal y administración contra ese presupuesto. Trabajar con los interesados para identificar y controlar los costos y beneficios totales dentro del contexto de los planes estratégicos y tácticos de TI y tomar medidas correctivas según sean necesarias. El proceso fomenta la sociedad entre TI y los interesados del negocio, facilita el uso efectivo y eficiente de recursos de TI y brinda transparencia y responsabilidad dentro del costo total de la propiedad, la materialización de los beneficios del negocio y el retorno sobre las inversiones en TI.

PO6 Comunicar las aspiraciones y la dirección de la gerencia. La dirección debe elaborar un marco de trabajo de control empresarial para TI y definir y comunicar las políticas. Un programa de comunicación continua se debe implantar para articular la misión, los objetivos de servicio, las políticas y procedimientos, etc., aprobados y apoyados por la dirección. La comunicación apoya el logro de los objetivos de TI y asegura la concientización y el entendimiento de los riesgos de negocio y de TI. El proceso debe garantizar el cumplimiento de las leyes y reglamentos relevantes.

PO7 Administrar los recursos humanos de TI. Adquirir, mantener y motivar una fuerza de trabajo para la creación y entrega de servicios de TI para el negocio. Esto se logra siguiendo prácticas definidas y aprobadas que apoyan el reclutamiento, entrenamiento, la evaluación del desempeño, la promoción y la terminación. Este proceso es crítico ya que las personas son activos importantes y el ambiente de gobierno y de control interno depende fuertemente de la motivación y competencia del personal.

PO8 Administrar la calidad. Se debe elaborar y mantener un sistema de administración de calidad, el cual incluya procesos y estándares probados de desarrollo y de adquisición. Esto se facilita por medio de la planeación, implantación y mantenimiento del sistema de administración de calidad, proporcionando requerimientos, procedimientos y políticas claras de calidad. Los requerimientos de calidad se deben manifestar y documentar con indicadores cuantificables y alcanzables. La mejora continua se logra por medio del constante monitoreo, corrección de desviaciones y la comunicación de los resultados a los interesados. La administración de calidad es esencial para garantizar que TI está dando valor al negocio, mejora continua y transparencia para los interesados.

PO9 Evaluar y administrar los riesgos de TI. Crear y dar mantenimiento a un marco de trabajo de administración de riesgos. El marco de trabajo documenta un nivel común y acordado de riesgos de TI, estrategias de mitigación y riesgos residuales acordados. Cualquier impacto potencial sobre las metas de la organización, causado por algún evento no planeado se debe identificar, analizar y evaluar. Se deben adoptar estrategias de mitigación de riesgos para minimizar los riesgos residuales a un nivel aceptable. El resultado de la evaluación debe ser entendible para los participantes y se debe expresar en términos financieros, para permitir a los participantes alinear los riesgos a un nivel aceptable de tolerancia.

P010 Administrar proyectos. Establecer un programa y un marco de control administrativo de proyectos para la administración de todos los proyectos de TI. El marco de trabajo debe garantizar la correcta asignación de prioridades y la coordinación de todos los proyectos. El marco de trabajo debe incluir un plan maestro, asignación de recursos, definición de entregables, aprobación de los usuarios, un enfoque de entrega por fases, aseguramiento de la calidad, un plan formal de pruebas, revisión de pruebas y revisión post-implantación después de la implantación para garantizar la administración de los riesgos del proyecto y la entrega de valor para el negocio. Este enfoque reduce el riesgo de costos inesperados y de cancelación de proyectos, mejora la comunicación y el involucramiento del negocio y de los usuarios finales, asegura el valor y la calidad de los entregables de los proyectos y maximiza su contribución a los programas de inversión en TI.

Adquisición e Implementación

AI1 Identificar soluciones automatizadas. La necesidad de una nueva aplicación o función requiere de análisis antes de la compra o desarrollo para garantizar que los requisitos del negocio se satisfacen con un enfoque efectivo y eficiente. Este proceso cubre la definición de las necesidades, considera las fuentes alternativas, realiza una revisión de la factibilidad tecnológica y económica, ejecuta un análisis de riesgo y de costo-beneficio y concluye con una decisión final de “desarrollar” o “comprar”. Todos estos pasos permiten a las organizaciones minimizar el costo para adquirir e implantar soluciones, mientras que al mismo tiempo facilitan el logro de los objetivos del negocio.

AI2 Adquirir y mantener software aplicativo. Las aplicaciones deben estar disponibles de acuerdo con los requerimientos del negocio. Este proceso cubre el diseño de las aplicaciones, la inclusión apropiada de controles aplicativos y requerimientos de seguridad y el desarrollo y la configuración en sí de acuerdo a los estándares. Esto permite a las organizaciones apoyar la operatividad del negocio de forma apropiada con las aplicaciones automatizadas correctas.

AI3 Adquirir y mantener infraestructura tecnológica. Las organizaciones deben contar con procesos para adquirir, implantar y actualizar la infraestructura tecnológica. Esto requiere de un enfoque planeado para adquirir, mantener y proteger la infraestructura de acuerdo con las estrategias tecnológicas convenidas y la disposición del ambiente de desarrollo y pruebas. Esto garantiza que exista un soporte tecnológico continuo para las aplicaciones del negocio.

AI4 Facilitar la operación y el uso. El conocimiento sobre los nuevos sistemas debe estar disponible. Este proceso requiere la generación de documentación y manuales para usuarios y para TI y proporciona entrenamiento para garantizar el uso y la operación correctos de las aplicaciones y la infraestructura.

AI5 Adquirir recursos de TI. Se deben suministrar recursos TI, incluyendo personas, hardware, software y servicios. Esto requiere de la definición y ejecución de los procedimientos de adquisición, la selección de proveedores, el ajuste de arreglos contractuales y la adquisición en sí. El hacerlo así garantiza que la organización tenga todos los recursos de TI que se requieren de una manera oportuna y rentable.

AI6 Administrar cambios. Todos los cambios, incluyendo el mantenimiento de emergencia y parches, relacionados con la infraestructura y las aplicaciones dentro del ambiente de producción, deben administrarse formalmente y controladamente. Los cambios (incluyendo procedimientos, procesos, sistema y parámetros del servicio) se deben registrar, evaluar y autorizar previo a la implantación y revisar contra los resultados planeados después de la implantación. Esto garantiza la reducción de riesgos que impactan negativamente la estabilidad o integridad del ambiente de producción.

AI7 Instalar y acreditar soluciones y cambios. Los nuevos sistemas necesitan estar funcionales una vez que su desarrollo se completa. Esto requiere pruebas adecuadas en un ambiente dedicado con datos de prueba relevantes, definir la transición e instrucciones de migración, planear la liberación y la transición en sí al ambiente de producción y revisar la post-implantación. Esto garantiza que los sistemas operacionales estén en línea con las expectativas convenidas y con los resultados.

Entrega y Soporte

DS1 Definir y administrar niveles de servicio. Contar con una definición documentada y un acuerdo de servicios de TI y de niveles de servicio, hace posible una comunicación efectiva entre la gerencia de TI y los clientes de negocio respecto de los servicios requeridos. Este proceso también incluye el monitoreo y la notificación oportuna a los participantes sobre el cumplimiento de los niveles de servicio. Este proceso permite la alineación entre los servicios de TI y los requerimientos de negocio relacionados.

DS2 Administrar los servicios de terceros. La necesidad de asegurar que los servicios provistos por terceros cumplan con los requerimientos del negocio, requiere de un proceso efectivo de administración de terceros. Este proceso se logra por medio de una clara definición de roles, responsabilidades y expectativas en los acuerdos con los terceros, así como con la revisión y monitoreo de la efectividad y cumplimiento de dichos acuerdos. Una efectiva administración de los servicios de terceros minimiza los riesgos del negocio asociados con proveedores que no se desempeñan de forma adecuada.

DS3 Administrar el desempeño y la capacidad. La necesidad de administrar el desempeño y la capacidad de los recursos de TI requiere de un proceso para revisar periódicamente el desempeño actual y la capacidad de los recursos de TI. Este proceso incluye el pronóstico de las necesidades futuras, basadas en los requerimientos de carga de trabajo, almacenamiento y contingencias. Este proceso brinda la seguridad de que los recursos de información que soportan los requerimientos del negocio están disponibles de manera continua.

DS4 Garantizar la continuidad del servicio. La necesidad de brindar continuidad en los servicios de TI requiere desarrollar, mantener y probar planes de continuidad de TI, almacenar respaldos fuera de las instalaciones y entrenar de forma periódica sobre los planes de continuidad. Un proceso efectivo de continuidad de servicios, minimiza la probabilidad y el impacto de interrupciones mayores en los servicios de TI, sobre funciones y procesos claves del negocio.

DS5 Garantizar la seguridad de los sistemas. La necesidad de mantener la integridad de la información y de proteger los activos de TI, requiere de un proceso de administración de la seguridad. Este proceso incluye el establecimiento y mantenimiento de roles y responsabilidades de seguridad, políticas, estándares y procedimientos de TI. La administración de la seguridad también incluye realizar monitoreos de seguridad y pruebas periódicas así como realizar acciones correctivas sobre las debilidades o incidentes de seguridad identificados. Una efectiva administración de la seguridad protege todos los activos de TI para minimizar el impacto en el negocio causado por vulnerabilidades o incidentes de seguridad

DS6 Identificar y asignar costos. La necesidad de un sistema justo y equitativo para asignar costos de TI al negocio, requiere de una medición precisa y un acuerdo con los usuarios del negocio sobre una asignación justa. Este proceso incluye la construcción y operación de un sistema para capturar, distribuir y reportar costos de TI a los usuarios de los servicios. Un

sistema equitativo de costos permite al negocio tomar decisiones más informadas respectos al uso de los servicios de TI.

DS7 Educar y entrenar a los usuarios. Para una educación efectiva de todos los usuarios de sistemas de TI, incluyendo aquellos dentro de TI, se requieren identificar las necesidades de entrenamiento de cada grupo de usuarios. Además de identificar las necesidades, este proceso incluye la definición y ejecución de una estrategia para llevar a cabo un entrenamiento efectivo y para medir los resultados. Un programa efectivo de entrenamiento incrementa el uso efectivo de la tecnología al disminuir los errores, incrementando la productividad y el cumplimiento de los controles clave tales como las medidas de seguridad de los usuarios.

DS8 Administrar la mesa de servicio y los incidentes. Responder de manera oportuna y efectiva a las consultas y problemas de los usuarios de TI, requiere de una mesa de servicio bien diseñada y bien ejecutada y de un proceso de administración de incidentes. Este proceso incluye la creación de una función de mesa de servicio con registro, escalamiento de incidentes, análisis de tendencia, análisis causa-raíz y resolución. Los beneficios del negocio incluyen el incremento en la productividad gracias a la resolución rápida de consultas. Además, el negocio puede identificar la causa raíz (tales como un pobre entrenamiento a los usuarios) a través de un proceso de reporte efectivo.

DS9 Administrar la configuración. Garantizar la integridad de las configuraciones de hardware y software requiere establecer y mantener un repositorio de configuraciones completo y preciso. Este proceso incluye la recolección de información de la configuración inicial, el establecimiento de normas, la verificación y auditoría de la información de la configuración y la actualización del repositorio de configuración conforme se necesite. Una efectiva administración de la configuración facilita una mayor disponibilidad, minimiza los problemas de producción y resuelve los problemas más rápido.

DS10 Administración de problemas. Una efectiva administración de problemas requiere la identificación y clasificación de problemas, el análisis de las causas desde su raíz y la resolución de problemas. El proceso de administración de problemas también incluye la identificación de recomendaciones para la mejora, el mantenimiento de registros de problemas y la revisión del estatus de las acciones correctivas. Un efectivo proceso de administración de problemas mejora los niveles de servicio, reduce costos y mejora la conveniencia y satisfacción del usuario.

DS11 Administración de datos. Una efectiva administración de datos requiere de la identificación de requerimientos de datos. El proceso de administración de información también incluye el establecimiento de procedimientos efectivos para administrar la librería de medios, el respaldo y la recuperación de datos y la eliminación apropiada de medios. Una efectiva administración de datos ayuda a garantizar la calidad, oportunidad y disponibilidad de la información del negocio.

DS12 Administración del ambiente físico. La protección del equipo de cómputo y del personal, requiere de instalaciones bien diseñadas y bien administradas. El proceso de administrar el ambiente físico incluye la definición de los requerimientos físicos del centro de datos (site), la selección de instalaciones apropiadas y el diseño de procesos efectivos para monitorear factores ambientales y administrar el acceso físico. La administración efectiva del ambiente físico reduce las interrupciones del negocio ocasionadas por daños al equipo de cómputo y al personal.

DS13 Administración de operaciones. Un procesamiento de información completo y apropiado requiere de una efectiva administración del procesamiento de datos y del mantenimiento del hardware. Este proceso incluye la definición de políticas y procedimientos de operación para una administración efectiva del procesamiento programado, protección de datos de salida sensitivos, monitoreo de infraestructura y mantenimiento preventivo de hardware. Una efectiva administración de operaciones ayuda a mantener la integridad de los datos y reduce los retrasos en el trabajo y los costos operativos de TI.

Monitoreo y Evaluación

ME1 Monitorear y evaluar el desempeño de TI. Una efectiva administración del desempeño de TI requiere un proceso de monitoreo. El proceso incluye la definición de indicadores de desempeño relevantes, reportes sistemáticos y oportunos de desempeño y tomar medidas expeditas cuando existan desviaciones. El monitoreo se requiere para garantizar que las cosas correctas se hagan y que estén de acuerdo con el conjunto de direcciones y políticas.

ME2 Monitorear y evaluar el control interno. Establecer un programa de control interno efectivo para TI requiere un proceso bien definido de monitoreo. Este proceso incluye el monitoreo y el reporte de las excepciones de control, resultados de las auto-evaluaciones y revisiones por parte de terceros. Un beneficio clave del monitoreo del control interno es proporcionar seguridad respecto a las operaciones eficientes y efectivas y el cumplimiento de las leyes y regulaciones aplicables.

ME3 Garantizar el cumplimiento regulatorio. Una supervisión efectiva del cumplimiento regulatorio requiere del establecimiento de un proceso independiente de revisión para garantizar el cumplimiento de las leyes y regulaciones. Este proceso incluye la definición de un estatuto de auditoría, independencia de los auditores, ética y estándares profesionales, planeación, desempeño del trabajo de auditoría y reportes y seguimiento a las actividades de auditoría. El propósito de este proceso es proporcionar un aseguramiento positivo relativo al cumplimiento de TI de las leyes y regulaciones.

ME4 Proporcionar gobierno de TI. El establecimiento de un marco de trabajo de gobierno efectivo, incluye la definición de estructuras, procesos, liderazgo, roles y responsabilidades organizacionales para garantizar así que las inversiones empresariales en TI estén alineadas y de acuerdo con las estrategias y objetivos empresariales.

Administración de bases de datos

Controles de bases de datos. Es crítico que la integridad y disponibilidad de la base de datos se mantenga; esto se asegura a través de los siguientes controles:

- Ø Establecer y obligar la definición de estándares, políticas y procedimientos
- Ø Establecer e implementar el respaldo de datos y los procedimientos de recuperación para asegurar la disponibilidad de la base de datos
- Ø Establecer los niveles necesarios de controles de acceso para los elementos de datos, tablas y archivos para prevenir acceso inadvertido o no autorizado
- Ø Establecer controles para asegurar que sólo el personal autorizado pueda actualizar la base de datos
- Ø Establecer controles para manejar los problemas de concurrencias, tales como múltiples usuarios deseando actualizar los mismos elementos de datos al mismo tiempo
- Ø 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.
- Ø 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
- Ø Ejecutar la optimización de las bases de datos para reducir espacio de disco no utilizado y verificar las relaciones de datos definidas
- Ø Seguir los procedimientos de reestructuración cuando se hagan cambios lógicos, físicos y de procedimientos
- Ø Usar herramientas de reporte de desempeño de las bases de datos para monitorear y mantener la eficiencia de la base de datos
- Ø 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.

<u>Revisiones de bases de datos</u>

Para la revisión de bases de datos deben considerarse los siguientes elementos:

- Diseño
  - 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.
  - 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).

Análisis del impacto del negocio. Un análisis de impacto en el negocio es el primer paso para desarrollar planes de continuidad del negocio y en especial de los planes de recuperación de desastres. Dentro de las diversas preguntas que deben formularse durante el análisis del impacto del negocio se encuentran:

- ¿Cuáles son los procesos de negocio? Esta pregunta es fundamental para determinar su criticidad para la organización.
- ¿Cuáles son los recursos de información críticos que están relacionados con los procesos de negocio críticos? Esta es la primera consideración debido a que una interrupción en un recurso de información no es un desastre en sí misma a menos que se relacione con procesos críticos de la organización.
- ¿Cuál es el periodo de tiempo crítico de recuperación para los recursos de información en los cuales los procesos de negocio deben ser resueltos antes de sufrir pérdidas significativas o inaceptables? En gran parte la longitud del tiempo de recuperación depende de la naturaleza del negocio o del servicio interrumpido.
- ¿Cuál es la clasificación de riesgos de los sistemas? Esto involucra una determinación de riesgos basados en el impacto derivado del periodo de tiempo crítico de recuperación como de la probabilidad de que ocurra una interrupción adversa.

Clasificación de las operaciones y análisis de criticidad de los sistemas. Una clasificación de riesgos de los sistemas típicos puede contener la siguiente clasificación:

Clasificación	Descripción
Crítico	Las funciones pueden realizarse sólo si las capacidades se reemplazan por otras idénticas No pueden reemplazarse por métodos manuales. Muy baja tolerancia a interrupciones. Muy alto costo de interrupción.
Vitales	Pueden realizarse manualmente por periodo breve Un poco más de tolerancia a interrupciones vs. Críticos. Costo de interrupción un poco más bajos, sólo si son restaurados dentro de un tiempo determinado (5 ó menos días).
Sensitivos	Las funciones pueden realizarse manualmente por un periodo prolongado, a un costo tolerable. Proceso manual difícil, requiriendo personal adicional.
No Críticos	Las funciones pueden interrumpirse por tiempos prolongados a un costo pequeño o nulo. No requiere de esfuerzos para actualizarse.

La Seguridad en una Base de Datos

Introducción

¿Por qué es importante?

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.

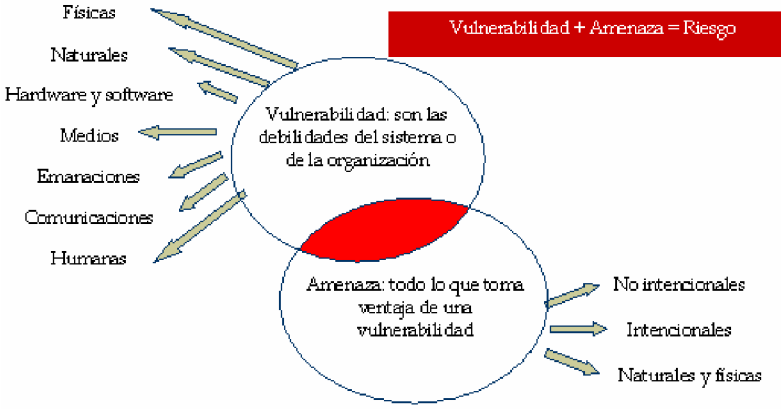
Los sistemas de bases de datos son sistemas extremadamente complejos y con gran dificultad para configurar y asegurar. La protección del sistema operativo y de los servicios de red en un servidor de bases de datos tiene una importancia crítica.

Secrecía y confidencialidad:

- Ø Los datos deben ser protegidos de liberación no autorizada
  - Ø Mediante recuperación directa o inferencia lógica
  - Ø Mediante lectura de usuarios no autorizados
- Ø Precisión, integridad y autenticidad;
  - Ø Los datos deben ser protegidos de modificación accidental o maliciosa (considerando incluso la inserción de datos falsos o la destrucción de los mismos)
  - Ø El origen de los datos debe ser verificable
- Ø Disponibilidad y recuperabilidad
  - Ø Los sistemas de bases de datos deben mantenerse operando y recuperarse en caso de pérdida de datos.

Tipos de Amenazas

- Ø Naturales
  - Ø Terremotos, incendios, ciclones
- Ø Políticas/Sociales
  - Ø Disturbios y huelgas
- Ø Físicas
  - Ø Fallas de energía, fallas del hardware, Bugs de software
- Ø Humanas
  - Ø No - Intencionales
    - Ø omisión, error humano
  - Ø Intencionales
    - Ø Usuarios autorizados que abusan de sus privilegios y autoridad
    - Ø Destrucción Física
    - Ø Virus
    - Ø Espionaje de mensajes
    - Ø Robo de Información
    - Ø Suplantación
    - Ø Ingeniería Social!!!
    - Ø Otros ataques de “hackers”



Vulnerabilidades

Consecuencias

- Ø Liberación inapropiada de información
- Ø Modificación inapropiada de datos
- Ø Negación de servicio
- Ø Robo o fraude

Vulnerabilidades

Los requerimientos mínimos necesarios que debe cumplir cualquier programa que se diga RDBM son:

1. Autenticación de Usuarios
   - Ø Necesidad de identificar de forma única a los usuarios de la base de datos.
   - Ø Puede ser realizada por:
     - Ø Sistema operativo
     - Ø DBMS
2. Protección de acceso impropio
   - Ø Permitir acceso a los objetos de la BD solamente a usuarios autorizados
   - Ø Granularidad más fina que en el caso de los sistemas operativos
3. Integridad física
   - Ø Sistema de respaldo y recuperación
   - Ø Uso de un log o Journal
   - Ø Instrucciones especiales para aplicar operaciones o cancelarlas
     - Ø Con puntos de inicio y de control (commit)
4. Administración y protección de datos sensibles
   - Ø Datos sensibles:
     - Ø Datos que no deben ser hechos públicos
   - Ø Factores que pueden hacer sensibles a los datos:
     - Ø Inherentemente sensible
     - Ø El valor mismo debe protegerse (sea declarado sensible o no)
     - Ø Proviene de una fuente sensible
     - Ø Debe protegerse la fuente de la que procede
     - Ø Se ha declarado sensible
     - Ø Se derivan de un registro o un atributo sensible
5. - Valor de la información.
   - Ø Puede haber bases de datos:
     - Ø Solamente con datos sensibles
     - Ø Solamente con datos públicos
     - Ø Con ambos tipos de datos

Integridad de datos

Definición

- Ø Expectativa de calidad de datos.
  - Ø Los datos tienen integridad si su calidad alcanza o supera los requerimientos de calidad que los usuarios esperan de ellos.
- Ø Protección contra la modificación impropia de datos.
- Ø Protección contra la modificación no autorizada de los datos.

Tipos de integridad

- Ø Integridad de dominio
  - Ø Requiere que un conjunto de valores sean válidos para una columna y especifica si se permiten valores nulos. Se implementa mediante la verificación de validez y se puede forzar restringiendo el tipo de datos, formato o rango de valores permitidos en una columna y en caso de omisión el valor de dafault
- Ø Integridad de entidad
  - Ø Requiere que todos los renglones de una tabla tengan un identificador único, conocido como llave primaria. El valor de una llave primaria puede cambiarse dependiendo del nivel de integridad requerido entre la llave primaria y otras tablas.
- Ø Integridad referencial
  - Ø Asegura que las relaciones entre la llave primaria (en una tabla referenciada) y las llaves foráneas (en una tabla referenciante) siempre se mantenga.
  - Ø Un renglón en una tabla referenciada no puede ser borrado y si una llave foránea hace referencia a ese renglón, tampoco la llave primaria puede modificarse.

Forzando la integridad

- Ø Integridad declarativa
  - Ø Los criterios de integridad se declaran en las definiciones de los objetos.
  - Ø Se implementa mediante el uso de restricciones declarativas que se definen directamente en tablas y columnas:
    - Ø Restricciones de llave
    - Ø Valores por omisión
    - Ø Valores de verificación
- Ø Integridad procedimental
  - Ø Criterios definidos en scripts.
  - Ø Se implementa mediante el uso de procedimientos almacenados y triggers.
  - Ø Se puede implementar en el cliente o en el servidor.

Control de Acceso

DAC

- Control de acceso discrecional (DAC)
  - Medio por el que se restringe el acceso a los objetos a usuarios específicos o grupos de usuarios.
  - El propietario de un objeto puede otorgar privilegios de acceso a otros usuarios sobre el mismo objeto, directamente o indirectamente.
- Privilegios
  - Medio por el que SQL implementa el DAC.
  - Se conceden privilegios por medio de una instrucción GRANT y se usan para especificar una acción permitida sobre un objeto específico

Vistas

- Los usuarios tienen acceso a la vista, pero no a la tabla base.
- Apropiadas para restricciones de columna
- Soportan herramientas de acceso y consultas ad-hoc
- Están especificadas en SQL – Son independientes del DBMS
- Se soporta la actualización

Mejores prácticas

Recomendaciones

- Usar un sistema de detección de intrusos, especialmente en servidores de bases de datos en línea y de alto riesgo.
- Cambiar las contraseñas de las cuentas creadas por omisión durante la instalación. Asignar contraseñas fuertes a las mismas.
- Deshabilitar las cuentas de invitado y las cuentas de demostración o ejemplo definidas durante la instalación. Eliminar estas cuentas en las bases de datos de producción.
- Mantener actualizado el DBMS con las versiones más recientes del software y de los parches de seguridad liberados por el fabricante del software.
- No permitir que las aplicaciones consulten o manipulen directamente la base de datos mediante instrucciones SELECT, INSERT, UPDATE o DELETE. Usar procedimientos almacenados en su lugar.
- Impedir que las aplicaciones acepten instrucciones de SQL de los usuarios y las ejecuten sobre la base de datos.
- Hacer que los usuarios consulten los datos mediante vistas en lugar de otorgarles acceso a las tablas base.
- Habilitar la auditoría de acceso al sistema operativo y al servidor de base de datos.
- Revisar el registro de auditoría buscando los eventos fallidos de acceso y buscar tendencias con la finalidad de detectar posibles intrusos.
- Monitorear cuidadosamente los registros (logs) de error y de eventos y disparar automáticamente alertas relacionadas con la seguridad y con errores. Proteger los archivos de registro (log) mediante permisos apropiados de sistema operativo.
- En ambiente de bases de datos distribuidas, eliminar el acceso a los servidores que no se utilicen. Utilizar cuentas de acceso con mínimos privilegios para los servidores relacionados.
- Almacenar los archivos utilizados para carga masiva de datos o por lotes (batch) en un directorio con los permisos apropiados. Eliminar los archivos una vez que hayan sido utilizados.
- Para asegurar la replicación de datos sobre Internet o sobre una red de área amplia (WAN), implementar una red privada virtual (VPN).
- Definir y aplicar una política de respaldo periódico. Almacenar los medios de respaldo en un lugar seguro. Realizar regularmente restauraciones de la base de datos a partir de los respaldos.

Seguridad en Cómputo

Introducción

Para comenzar es preciso definir los conceptos de “Sistema Operativo Linux” de tal forma que sea posible la comprensión de los siguientes temas. Linux es un sistema operativo que administra los recursos de la computadora tales como memoria, espacio en disco,

procesamiento y dispositivos periféricos, instalado en hardware CISC y RISC (utilizado para servidores de alto rendimiento).

Actualmente el sistema operativo Linux es uno de los principales sistemas operativos usados en grandes organizaciones, tales como bancos, organizaciones gubernamentales y compañías internacionales, entre otras, donde se usan bases de datos, sistemas de seguridad, sistemas de comunicación de área amplia para proveer servicios en Internet – Intranet - Extranet.

Linux ha incorporado en su diseño las siguientes características:
Ø Seguridad implícita de la información
Ø Modelo de sistemas abiertos (facilidad de integración en sistemas complejos)
Ø Manejo de grandes cantidades de transacciones
Ø Multiproceso y multiusuario
Ø Software libre y facilidad de desarrollo

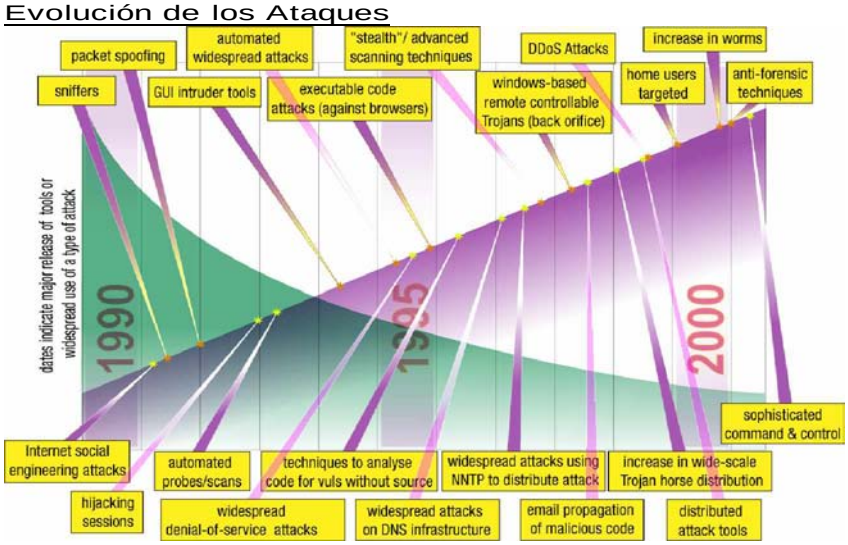
El tema de seguridad ha cobrado gran importancia debido a la serie de ataques informáticos que se ha venido registrando desde mediados de los años 80, ataques que se han sofisticado cada día debido a los avances tecnológicos que se han venido desarrollando y a la facilidad de uso de la Internet, la super-red de computadoras.

Un ataque informático lo podemos definir como el aprovechamiento de las vulnerabilidades de un sistema para explotarlas comprometiendo la información contenida en dichos sistemas. El sufrir un ataque no solo conlleva pérdida de información, daño a equipos, rediseño de procesos, inversiones millonarias, sino también la pérdida de la imagen, credibilidad y de la confianza en las organizaciones, lo cual muchas veces es más catastrófico que las pérdidas monetarias.

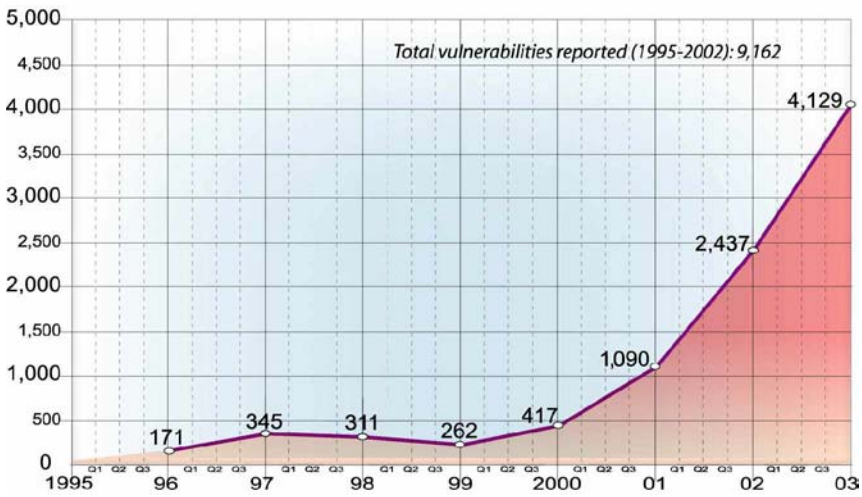
Historia de la Seguridad

Año	Autor	Descripción
1986	Clifford Stoll	Rastreo de intrusos que trataban de violar sistemas de computo de los Laboratorios Lawrence Berkeley.
1986	Captain Midnight	Mensaje satelital enviado a 8 millones de usuarios de la HBO.
1988		Gusano que se introdujo en miles de equipos de universidades y gobierno.
1988		Virus Viernes 13 infecto cientos de computadoras borrando información.
1991		Borrado accidental de la nueva Constitución de Colombia.
1994	Kevin Mitnick	Robo de software y tarjetas de crédito a través de ingeniería social y ataques IP-Spoofing.
1996		Alteración de la pagina web de la fuerza aérea de EUA
1998		Alteración de la pagina web de la fuerza aérea del Departamento de Justicia
1998	X-Ploit	En México: Secretaría de Hacienda y Crédito Público, INEGI, Secretaría de Salud y Senado de la República

Evolución de los Ataques



Vulnerabilidades Reportadas



Sistemas Abiertos

Un sistema abierto es aquel que permite una fácil integración de sus componentes basándose en estándares.

El estándar concerniente con la estructura del sistema de comunicación es el llamado OSI (Open System Interconexión) desarrollado por la ISO (International Standards Organization). La mayoría de las aplicaciones de sistemas abiertos y distribuidos se basan en el modelo cliente/servidor.

Las organizaciones internacionales dedicadas al desarrollo de estándares son:
- La ISO y el IEEE (the Institute of Electrical and Electronic Engineer) producen estándares para uso de fabricantes de computadoras
- La ITU-T que define estándares para la conexión de equipos a través de redes públicas y privadas

Las características que permiten a un sistema operativo la integración de sistemas abiertos son:
- Manejo de la memoria y de la concurrencia de procesos
- Permisos de usuarios
- Independencia del hardware de servidor
- Administración de fallas
- La adopción del modelo TCP/IP (Aunque el estándar TCP/IP no fue desarrollando basado en ISO, contiene toda la funcionalidad del ISO en sus capas).

Que es la seguridad en Cómputo

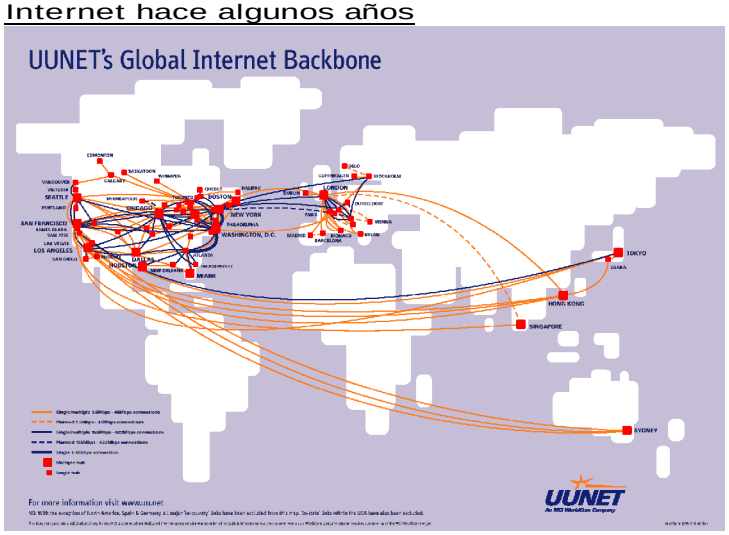
La Seguridad en Cómputo son mecanismos tecnológicos que protegen los sistemas de cómputo y todo lo asociado con ellos (edificios, impresoras, cableado, diskettes, etc.)
- Hay que distinguir dos conceptos importantes de seguridad:
  - Seguridad en Cómputo: enfocado a sistemas de cómputo y redes de datos.
  - Seguridad de la Información: enfocado al tratamiento y uso de la información, involucra sistemas de cómputo, redes de datos, gente y procesos.

¿ Que es la seguridad de la información ?
- La seguridad no es la meta de la organización
- La seguridad no es un aspecto técnico más a considerar
- La seguridad no se resuelve con un producto
- La seguridad de la información es el sistema de gestión que protege la información de una organización para asegurar la continuidad de negocio.

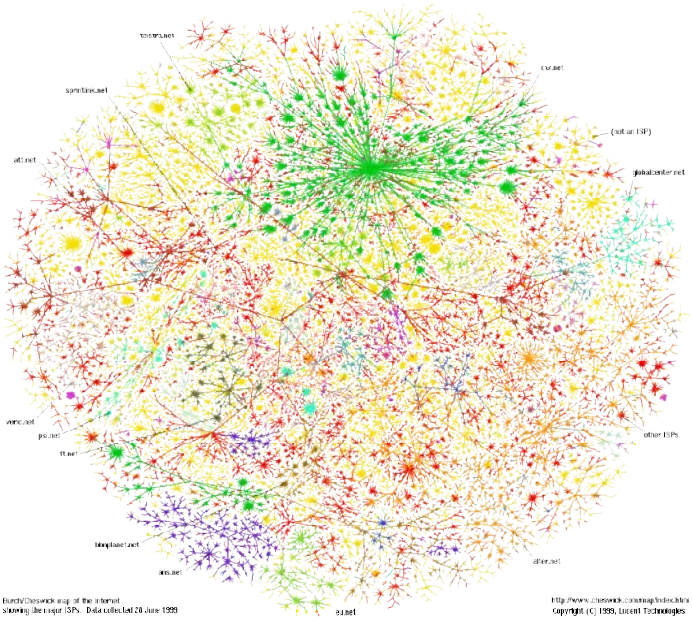
SISTEMAS 100% SEGUROS

“El único sistema totalmente seguro es aquel que está apagado, desconectado, guardado en una caja fuerte de titanio, encerrado en un bunker de concreto, rodeado por gas venenoso y cuidado por guardias muy bien armados y muy bien pagados. Aún así, no apostaría mi vida por él.” (Eugene Spafford)

Internet hace algunos años



Internet Hoy



Las redes de computadoras cambiaron dramáticamente las reglas de la seguridad. Unix fue originalmente desarrollado en un ambiente donde las computadoras estaban conectadas en un laboratorio pequeño sin contacto con el exterior y las redes actuales conectan miles de máquinas y millones de usuarios en todo el mundo, por esta razón cada uno de nosotros se confronta con los problemas de la seguridad directamente, ¿como puede una persona imaginar que un estudiante al otro lado del mundo atacará su computadora, leerá sus archivos y borrará su información?

Las redes han hecho de la seguridad una preocupación diaria y no solo un aspecto teórico, el acceso a redes públicas y privadas es un riesgo que se considera importante para todas las organizaciones actuales; sin embargo, las redes son necesarias porque permiten la comunicación, el compartir recursos, el intercambio de información y en algunos casos minimiza costos en contraste con los sistemas de comunicación tradicionales.

Es posible minimizar riesgos en los sistemas operativos realizando un “hardening”, auditorias informáticas, instalar herramientas y contar con un buen administrador.

Pero también los fabricantes de hardware para redes de datos se han preocupado cada vez más sobre los temas de seguridad ya que como cualquier equipo, también el hardware de red cuenta con usuarios, sistemas operativos y aplicaciones de administración, por lo que cada fabricante incluye características de seguridad en sus productos. Podemos encontrar que por ejemplo Nortel y Cisco incluyen módulos de seguridad en sus sistemas operativos y de hecho Cisco cuenta con productos tales como Firewalls, VPN´s, cifrado con clientes, entre otros productos, como módulos de los routers.

En las redes de alta velocidad se utiliza el concepto de QoS que incluye aspectos de seguridad para protección de la información, además los proveedores de redes públicas y privadas mantienen contratos con cláusulas de protección de la información.

Existe una gran competencia de los productos de seguridad que son instalados sobre la red de transmisión de datos y de los que son instalados a nivel servidor, es común perderse y confundirse con el mar de información y de supuestas ventajas de unos productos sobre otros.

Otra Clasificación del tipo de amenaza

Por el tipo de afectación en el sistema

- Pasivas
  - Intercepción (Confidencialidad)
    - Captura del contenido de los mensajes
    - Análisis de tráfico
- Activos
  - Negociación del servicio (disponibilidad)
  - Modificación o Destrucción (integridad)
  - Enmascaramiento y “Replay” (Autenticidad)

Vulnerabilidades (Top 10 ISS)

- 01. Ataques de negación del servicio
  - Stacheldraht
  - FunTime Apocalypse
- 02. Cuentas débiles
  - Cuentas por omisión (firewalls y routers)
  - Passwords nulos (Administrator/root)
  - SNMP con comunidades públicas/privadas
- 03. IIS (Microsoft Internet Information Server)
  - RDS (Remote Data Service)
  - HTR, Malformación del encabezado
  - Errores en CGIs
  - Metacaracteres PHP3 y lectura PHP
- 04. Bases de datos abiertas
  - . Passwords por omisión de Oracle
  - . Desbordamiento de buffer Xp_sprintd en SQL Server
  - . Ejecución de comandos Xp_cmdshell en SQL Server
- 05. Aplicaciones de comercio electrónico
  - . NetscapeGetBo
  - . FrontpagePwdAdministrators
- 06. Correo electrónico abierto
  - . Ataque Sendmail pipe
  - . SendmailMIMEbo
- 07. Compartir recursos
  - . NetBios
  - . NFS
- 08. RPC (Remote Procedure Call)
  - . Rpc.cmsd, rpc-statd, sadmin, amd, Mountd
- 09. BIND
  - . BIND nxt
    - Respuesta servidor a servidor
    - Manejo de desbordamiento de buffers y otros.
  - . BIND qinv
    - Bandera de compilación encendida por omisión
    - Desbordamiento de buffer activado
    - Peticiones del cliente al servidor
- 10. LINUX
    - IMP BO, Qpopper BO, Sobreescritura de Stack
    - Exploits de scripts comunes
    - Estándares débiles en código de programación

El mundo underground

- 2600 Magazine (Emmanuel Goldstein).
  - Revista técnica.
  - Principios: No robo, no fraude, no derechos de autor, libertad.
  - En 1985 tuvo problemas con el FBI por publicación de información para crackear.
- Revistas:
  - Rampart Magazine publico como funcionan las “Cajas azules”.

- Ø Phrack Magazine
- Ø Organizaciones:
  - Ø The Hacker's Defense Fund.
  - Ø Asociación de HACKERS españoles.
  - Ø X-ploit Team (México)
  - Ø Mex-hack (México)

Modelos de Seguridad

Los estándares internacionales BS7799 e ISO 7498-2 no diferencian tipos de seguridad; sin embargo, la norma BS7799 indica cuales son todos los aspectos que debe cubrir la auditoría informática. Políticas de seguridad de la información:

- Ø Seguridad de información en comunicaciones y administración de las operaciones
- Ø Seguridad física y ambiental de la información
- Ø Control de acceso
- Ø Desarrollo y mantenimiento de sistemas considerando seguridad informática
- Ø Gestión de la continuidad de operación
- Ø Aspectos legales y contractuales de la seguridad de información
- Ø Controles de seguridad de información para el Personal
- Ø Control y clasificación de bienes y de información
- Ø Infraestructura de seguridad de información

Cualidades de la información segura

- Ø Validez y Legitimidad. Es la forma de verificar quien envio los datos cuando fueron enviados y recibidos.
- Ø Confidencialidad. Acceso de información por entidades con acceso autorizado.
- Ø Integridad y precisión. Suficiencia y validez de la información. No debe ser modificada de forma maliciosa o accidental.
- Ø Disponibilidad. Acceso a la información cuando esta es requerida, así como la habilidad de recuperarse rápida y completamente si un evento ocurre

Las características de autenticación, confidencialidad, disponibilidad, integridad y el no repudio, están definidas en el estándar ISO 7498-2 en sus “Servicios de Seguridad”.

La Norma BS7799 define la confidencialidad, integridad, disponibilidad, autenticidad, los controles a considerar y los aspectos que debe cubrir una Auditoria de Seguridad para obtener un Sistema de Gestión de Seguridad de la Información.

La herramienta básica para cumplir las condiciones anteriores son las técnicas de cifrado también llamadas “criptográficas”, en particular los métodos de cifrado:

- Ø Simétrico. Usan una misma clave secreta para cifrar y descifrar
- Ø Asimétrico. Cada usuario tienen una pareja de claves, una pública y otra privada, lo que cifra con una de las claves solo se puede descifrar con la otra

Los métodos de cifrado simétrico, por ejemplo el sistema DES, usan una misma clave para cifrar y descifrar. Se requiere que los dos interlocutores compartan una clave secreta y de longitud suficientemente grande. Este esquema es poco adecuado cuando una parte establece comunicaciones ocasionales con otras con las que no tenía una relación previa ya que antes de poder establecer cada comunicación sería necesario intercambiar previamente por algún procedimiento seguro la clave que se va a utilizar para cifrar y descifrar la comunicación.

Los métodos de cifrado asimétrico, también llamado de clave pública, por ejemplo el sistema RSA, usan parejas de claves con la propiedad de que lo que se cifra con una de las claves de una pareja solo se puede descifrar con la otra clave de la pareja.

Una persona solo necesita tener una pareja de claves que puede utilizar para comunicarse de forma segura con cualquier otra persona que disponga a su vez de otra pareja de claves. Cada persona hace pública una de sus claves (clave o llave pública) y mantiene en secreto la otra (clave o llave privada).

Para enviar un mensaje de forma confidencial a un destinatario basta cifrarlo con la clave pública de ese destinatario, así solo él podrá descifrarlo mediante la clave privada que mantiene en secreto. No es necesario que el remitente y el destinatario intercambien previamente ninguna clave secreta.

Para evitar posibles suplantaciones de identidad, es necesario contar con una tercer parte fiable que acredite de forma fehaciente cual es la clave pública de cada persona o entidad,

esta es la función de las autoridades de certificación. La infraestructura necesaria para el uso de los sistemas de clave pública, incluyendo las autoridades de certificación se llama Infraestructura de Clave Pública (PKI – Public Key Infraestructure)

El primer paso es reconocer y entender el problema, ¿se necesita un sistema de seguridad de la información? ¿por qué y para que?. Se tienen que responder las siguientes preguntas:
- Ø ¿Qué se quiere proteger? identificación de bienes.
- Ø ¿Contra qué se quiere proteger? Identificación de vulnerabilidades y amenazas.
- Ø Cuánto tiempo, dinero y esfuerzo se está dispuesto a invertir para protegerlo. Identificación de impactos y riesgos.

Modelos de Seguridad
- Ø ¿Qué papel juega la información en el éxito de tus negocios/organización?
- Ø ¿Qué pasaría si no puedes acceder a tu información electrónica? ¿cuál sería el impacto?
- Ø ¿Qué tipo de información es más crítica para tu negocio/organización?
- Ø ¿Qué pasaría si tus competidores o enemigos obtuvieran está información?
- Ø ¿Qué impacto tendría para tu organización si de repente un periódico pública un artículo de cómo te “hackearon”?
- Ø ¿Que pasaría si NO proteges tu información?

La seguridad no puede garantizarse al 100% por lo que el mejor sistema de seguridad es el que cuenta con sistemas de defensa (para no ser atacado) y de recuperación (en caso de ser atacado) para minimizar los daños.

Una respuesta rápida en caso de tener un incidentes de seguridad no es solamente software y hardware, más bien son políticas, controles y procedimientos.

Tipos de Seguridad
Existen 2 tipos de seguridad, la seguridad física y la seguridad lógica:
- Ø Seguridad. Física: Se refiere a protección de la gente, inmuebles, medios de transmisión y almacenamiento, equipo de cómputo y comunicaciones.
  - Ø La forma de proteger los bienes físicos (tangibles) es con programas de mantenimiento, procedimientos de logística y mecanismos físicos en general.
- Ø Seguridad. Lógica: Se refiere a protección del software, datos e información, procedimientos y en general bienes intangibles.
  - Ø La forma de proteger los bienes lógicos (tangibles) es con programas de software, capacitación, controles y políticas.

Políticas de Seguridad

Las políticas de seguridad son un conjunto de reglas y prácticas que regulan como una organización administra, protege y distribuye información sensible. Es la forma en como un sistema provee confianza.

Una política de seguridad está escrita en términos de sujetos y objetos. Un sujeto es algo activo en el sistema, por ejemplo: usuarios, procesos y programas. Un objeto es aquello sobre el cual el sujeto aplica una acción, por ejemplo: archivos, directorios, dispositivos, etc. Una vez identificados los sujetos y objetos de un sistema, deben existir políticas que determinen que acciones le están permitidas a un sujeto realizar sobre un objeto.

Las funciones de la política son:
- Ø Proveer una decisión unitaria que se aplique a todas las situaciones similares
- Ø Orientar en forma clara hacia donde deben dirigirse todas las actividades de un mismo tipo
- Ø Establecer una manera consistente de tratar a la gente
- Ø Proveer un lineamiento que facilite la toma de decisiones en actividades rutinarias
- Ø Dar a conocer lo que la Dirección desea que se haga en cada situación definida

Las políticas ayudan a evitar lentitud, defectos y sobre, todo pérdida de tiempo en las principales actividades y procesos de la organización.

Al estar relacionadas con las personas, las políticas surgen en todas las actividades:
- Ø Procesos. En procesos de planeación, de atención a clientes, de recepción de materiales, de pago a proveedores, de elaboración y entrega de pedidos, de contratación y evaluación de personal, seguridad en procesos, etc.
- Ø Sistemas. En sistemas de calidad, de control, de investigación y desarrollo, de mejora, de cómputo, seguridad en sistemas, etc.

- Ø Planes. En planes para controlar y reducir costos, eliminar desperdicios, para desarrollar proveedores, para mejorar la comunicación, para impartir capacitación, de contingencia y recuperación, etc.
- Ø Mejoras. En mejoras de procesos, sistemas, planes, estrategias, proyectos, etc.
- Ø Conflictos. En conflictos con clientes, proveedores, otras áreas y niveles jerárquicos dentro de la empresa, etc.

Características de una política
- Ø Establece lo que la Dirección quiere o prefiere que se haga
- Ø No dice como proceder (eso lo dice el procedimiento)
- Ø Refleja una decisión directiva para todas las situaciones similares
- Ø Ayuda a las personas de nivel operativo, a tomar decisiones firmes y congruentes con la Dirección.
- Ø Tiende a darle consistencia a la operación
- Ø Es un medio para que a todos se les trate equitativamente
- Ø Orienta las decisiones operativas en la misma dirección
- Ø Ayuda a que todas las actividades de un mismo tipo, tomen la misma dirección
- Ø Les quita a los ejecutivos la molestia de estar tomando decisiones sobre asuntos rutinarios

Hay dos aspectos indispensables para poder implantar con éxito las políticas de seguridad:
- Ø Que el manual -y por ende las políticas- sea aprobado oficialmente por la mesa directiva de una organización.
- Ø Que el manual sea revisado y actualizado periódicamente.
- Ø Que todo el personal de la organización lo lea, entienda y se comprometa a acatarlo

El Manual de Políticas de Seguridad tiene las siguientes ventajas:
- Ø Multiplican y aceleran el conocimiento (y la experiencia) de la gente
- Ø Documentan la tecnología de una organización
- Ø Aseguran la calidad de los servicios (y productos)
- Ø Evitan estrés y desperdicio organizacional
- Ø Entrenan rápidamente al personal
- Ø Incrementan la competitividad de una organización

La desventaja de las políticas es:
- Ø Para su implementación en cualquier organización encuentran resistencia al cambio
- Ø Si no están bien elaboradas pueden causar confusión y ser parte de la burocracia
- Ø Si no se actualizan se convierten en obsoletas

Estructura de una política escrita:
- Ø Nombre de la política
- Ø Objetivo
- Ø Alcance
- Ø Bitácora de revisiones y modificaciones
- Ø Procedimiento
- Ø Anexos

Como asegurar que una política se cumpla:
- Ø Con mucha disciplina de parte de la Dirección para respetarla
- Ø Vigilar que se cumpla cabalmente en todos los niveles jerárquicos
- Ø Antes de aprobarla, analizar con mucho cuidado sus pros y contras. Seleccionar la más adecuada.
- Ø Involucrar en el diseño a los usuarios, para obtener sus puntos de vista y compromisos correspondientes.
- Ø Hacer una difusión formal y adecuada a todo el personal relacionado, a través de cartas, memorándum, boletines, pláticas, juntas, minutas, planes y manuales
- Ø Lograr que la gente involucrada comprenda claramente los beneficios y el porque de la política
- Ø Medir, evaluar y difundir los resultados de los indicadores asociados a dicha política
- Ø Realizar periódicamente auditorias para verificar en base a evidencia específica el apego a las políticas
- Ø Dar reconocimiento a aquellas personas o departamentos con mayores resultados

La política solamente es un medio para alcanzar un fin, la política por sí misma no es un fin. Excepcionalmente las políticas permiten excepciones.

Capítulo VI. Modelado Orientado a Objetos

Metodologías orientadas a objetos

Introducción

Componentes de Sistema de información
- Ø Software y Hardware
- Ø Personas
- Ø Documentación
- Ø Procesos (Reglas de negocio)
- Ø Bases de Datos (información)

Características del Software
- Ø Intangible
- Ø Complejo
- Ø No se gasta pero llega a ser obsoleto
- Ø Es una parte muy importante de los sistemas, más costoso que el hardware
- Ø La industria del software sigue en crecimiento

Problemática en el proyecto de desarrollo de software
- Ø Mala estimación y dimensionamiento de los proyectos
- Ø Fallas en el manejo de riesgos
- Ø Complejidad del software
- Ø La definición de requerimientos es mala y no se documentan
- Ø Los sistemas legados deben ser mantenidos por otras personas
- Ø Diseño y arquitectura inadecuados
- Ø No se realiza un modelado detallado de las características y componentes del software

Estadísticas en el desarrollo de software
- Ø El promedio del costo sobrepasado es de 189%
- Ø El tiempo de retraso promedio es de 222%
- Ø El promedio de fallas según las expectativas iniciales es de 61%
- Ø Proyectos exitosos 28%

Qué es un modelo
- Ø Es una representación de algo real
- Ø Un modelo capta los aspectos importantes de lo que estamos representando, desde cierto punto de vista y simplifica u omite el resto
- Ø Los modelos pueden ser físicos, gráficos, matemáticos

Por qué modelar
- Ø Es difícil comprender el todo
- Ø Permite dividir un problema complejo en problemas menores
- Ø Visualizar un sistema desde varias perspectivas
- Ø Entender y dimensionar el problema
- Ø Comprender y probar la solución
- Ø Abstraer las características, componentes y estructura de algo por construir
- Ø Detectar fallas, inconsistencias y prever cambios
- Ø Documentación del proyecto
- Ø Es menos costoso construir un modelo que un sistema
- Ø Comunicación entre el equipo de desarrollo y con los usuarios

Aplicaciones del Modelado
- Ø Construcción
- Ø Automotriz
- Ø Aeronáutica

- Eléctrica y electrónica
- Telecomunicaciones
- Administración
- Software

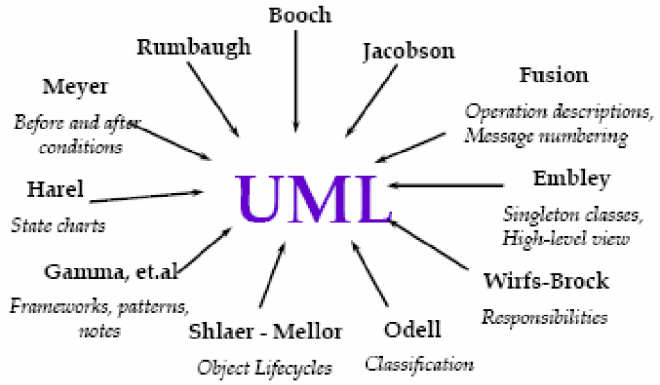
Con qué modelamos el software

- Análisis Estructurado (Yourdon)
- Metodología Jackson
- Técnicas de Diseño y Análisis Estructurado (SADT)
- UML (Orientación a objetos). permite que los usuarios, analistas, diseñadores y programadores de software comprendan un lenguaje de modelado estándar

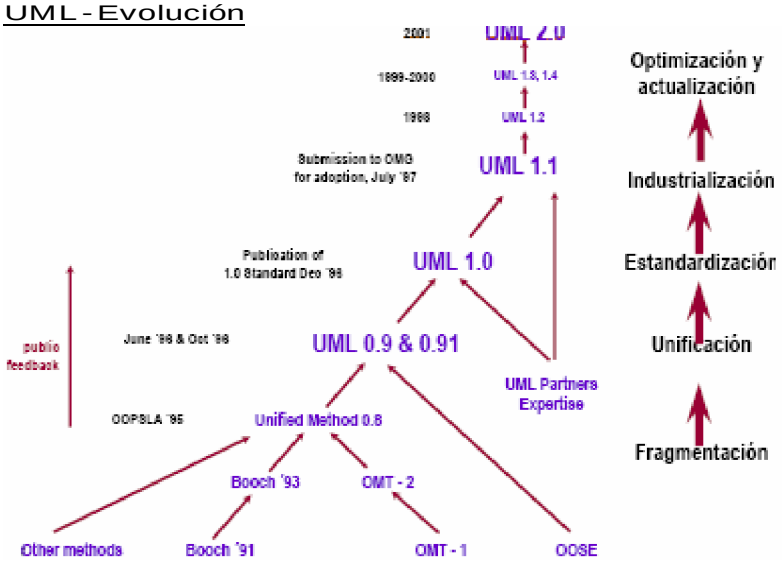
Qué es UML

- El Lenguaje de Modelado Unificado (UML) es un lenguaje de modelado visual usado para especificar, visualizar, construir y documentar artefactos de un sistema de software.
- Fue desarrollado en un esfuerzo para simplificar y consolidar las notaciones de desarrollo orientado a objetos que habían surgido, principalmente de Booch, Rumbaugh y Jacobson.
- Apoyado por el OMG (Object Management Group), Rational Software, Microsoft, Hewlett-Packard, Oracle, Texas Instruments, MCI Systemhouse y otros.
- En 1996 surge la versión 1.0.
- En 1997 surge el OMG y adopta los estándares relacionados.
- La versión más reciente de UML es la 2.0.

UML - Autores



UML - Evolución



UML

- 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.
- Se considera un estándar de facto. Es aquel patrón o norma que se caracteriza por no haber sido consensuada ni legitimada por un organismo de estandarización. Por el contrario, se trata de una norma generalmente aceptada y ampliamente utilizada por iniciativa propia de un gran número de interesados.

- UML es un lenguaje (de modelado) y no un método
- No es un proceso de software
- Incluye una serie de diagramas; especifica la notación para representarlos pero no describe cómo crearlos
- Define una notación expresiva y consistente
- Facilita la comunicación con otros y mayor involucramiento de los usuarios/clientes
- Permite detectar omisiones o inconsistencias
- Existen herramientas (Rational Rose) en el mercado para modelar, generar código y documentación a partir de UML
- Permite modelar y documentar la arquitectura de una aplicación.
- Reduce la complejidad.
- Permite la reutilización.
- Organización del UML:
  - Elementos
  - Relaciones
  - Diagramas

Generalidades del Paradigma OO

Antecedentes del paradigma OO

- 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:
  - POO. Método de programación en el que los programas se organizan como colecciones cooperativas de objetos
  - DOO. Método de diseño que abarca el proceso de descomposición OO y una notación para describir los modelos lógico, físico, estático y dinámico del sistema
  - AOO. Método de análisis que examina los requisitos desde la perspectiva de los objetos que se encuentran en el dominio del problema

Qué es el paradigma OO

Paradigma

- Forma o patrón de pensamiento
- Establece reglas y límites.
- Es una forma de resolver un 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.

Evolución de la Orientación a Objetos

70's - Métodos de desarrollo para lenguajes de programación como Cobol y Fortran

80's - Análisis estructurado y diseño estructurado (Yourdon, Ward, DeMarco, etc.)

1967 - Se desarrolla Simula 67, el primer lenguaje reconocido como OO

80's - Surgen y evolucionan otros lenguajes como: Smalltalk, Objetive C, C++, Eiffel, CLOS. Primeros métodos de desarrollo orientado a objetos

90's Unificación de métodos orientados a objetos

Ventajas del paradigma OO

- Representación de la realidad, sólo lo más importante.
- Es una herramienta de comunicación.
- Permite planear y administrar un proyecto adecuadamente.
- Aumento de la calidad de los productos de trabajo
- Reutilización
- Desarrollo más rápido
- Mayor calidad
- Facilidad de mantenimiento

- Modularidad
- Escalabilidad
- Mejores estructuras de información
- Incremento en adaptabilidad
- Desarrollo más flexible
- Librerías comerciales disponibles

Aplicaciones
- Sistemas basados en GUI
  - La OO facilita el diseño e implementación de sistemas con Interfaces Gráficas de Usuario (GUI)
- Los métodos OO permiten desarrollar sistemas inmersos y de tiempo real con mayor calidad y flexibilidad
- Comercio electrónico
  - Las aplicaciones de comercio electrónico requieren robustez y estabilidad que la tecnología de objetos proporciona
- JAVA

Objeto

Objeto
- Es un concepto, abstracción o cosa con límites bien definidos y significado para una aplicación.
- Representa un elemento identificable con ciertas características (atributos) y que puede realizar un conjunto de acciones (operaciones o métodos).
- Es algo que tiene:
  - Estado
  - Comportamiento
  - Identidad
  - Un objeto es una instancia de una clase
- De manera informal, un objeto representa una entidad ya sea
  - Física
  - Conceptual
  - o de software
- En la vida diaria un objeto es una cosa tangible o conceptual
- En análisis, un objeto es una entidad conceptual del dominio del problema
- En diseño, un objeto es una entidad detallada con límites bien definidos que representa una parte de la solución
- En código, un objeto es un bloque de código.
- La pieza fundamental que combina tanto estructura como comportamiento
- En contraste con la programación convencional, el elemento donde la estructura de datos y el comportamiento están estrechamente relacionados

Objeto
Características
Comportamiento

cliente1 : Cliente
nombre= “Edgar” apPaterno= “Islas” apMaterno=“Espinoza” edad=20
registrar() modificar() eliminar()

cliente2 : Cliente
nombre= “David” apPaterno= “Gómez” apMaterno=“García” edad=54
registrar() modificar() eliminar()

Abstracción
- Es la propiedad que permite representar las características esenciales de un objeto.
- Nos permite omitir algunas propiedades y acciones de un objeto y dejar sólo aquellas que nos interesan para una situación en particular.

Clasificación

Agrupar los objetos con características comunes en clases.

Encapsulamiento
- En general, el encapsulamiento hace referencia a cualquier tipo de ocultamiento.
- Propiedad que permite asegurar que el contenido de la información de un objeto está oculta al mundo exterior; protege los datos.
- La implementación es oculta del usuario.
- Toda la información (atributos y métodos) son almacenados dentro del objeto. La información puede ser manipulada a través de operaciones.

Clase

Clase

- Es una descripción de un grupo de objetos con propiedades comunes (atributos), comportamiento común (operaciones) y relaciones comunes con otros objetos (asociaciones y agregaciones).
- Es una definición abstracta de un objeto; sirve como una plantilla para crear objetos.
- Se pueden utilizar para representar software, hardware o puramente conceptuales.
- Una clase es una abstracción en la que ella enfatiza características relevantes y suprime otras características
- Son los bloques de construcción más importantes de cualquier sistema
- Las clases bien estructuradas están bien delimitadas y forman parte de una distribución equilibrada de responsabilidades en el sistema.
  - En análisis, una entidad conceptual del dominio del problema
  - En diseño, una entidad detallada que representa una parte de la solución
  - En código, es un bloque de código

Perspectivas. Una clase es una descripción de un grupo de objetos con los diagramas de clases se pueden construir desde tres perspectivas:

- Conceptual. Representa los conceptos del dominio que se está estudiando. Se dibujan sin importar el software con que se implementarán por lo que son independientes del lenguaje. Pueden ubicarse en el contexto del negocio (modelado de negocio) o del sistema.
- Especificación. Enfocados al software en su filosofía pero no en su implementación.
- Implementación. Se expone por completo la implementación.

Cómo identificar las clases en un problema. Las clases surgen de terminología de un área de conocimiento. Hay que prestar atención a:

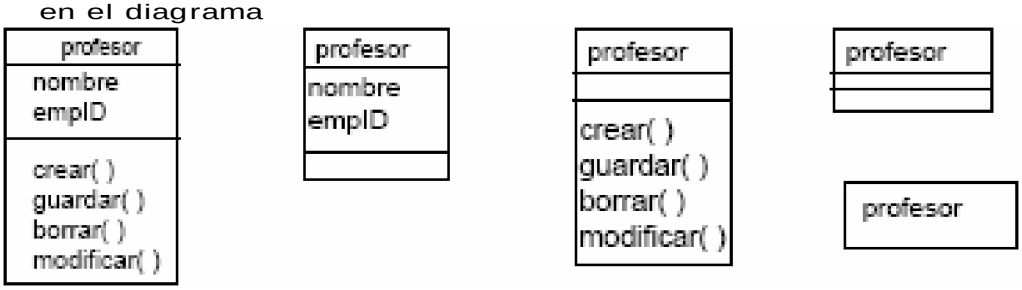
- Los sustantivos (cosas, personas, hechos) se pueden convertir en clases del modelo
- Verbos que pueden hacer operaciones de las clases o en clases por sí mismas
- Los atributos de una clase también se pueden identificar por sustantivos específicos que pueden tomar algún valor

Cómo nombrar las clases

- El nombre de una clase debe ser un nombre singular que caracterice de la mejor forma a la abstracción
  - La dificultad en el nombramiento de una clase puede indicar que una abstracción está pobremente definida
  - Los nombres deben venir directamente del vocabulario del dominio
- Una guía de estilo debe dictar convenciones de nombres para clases. Ejemplo:
  - Las clases se nombran usando sustantivos singulares
  - Los nombres de clases empiezan con una letra minúscula
  - No se usan palabras subrayadas. Los nombres compuestos de palabras múltiples se ponen juntas y la primera letra de cada palabra adicional se escribe en mayúscula. Ejemplo: alumno, profesor, sistemaCobro

Notación. Una clase se comprende de tres secciones

- La primera sección contiene el nombre de la clase
- La segunda sección muestra la estructura (atributos)
- La tercera sección muestra el comportamiento (operaciones)
- Las secciones segunda y tercera pueden suprimirse si no es necesario que sean visibles en el diagrama



Atributos. Un atributo es una propiedad de una clase identificada con un nombre, que describe un rango de valores que pueden tomar las instancias de la propiedad

Nombrado de atributos. Un atributo se puede especificar más indicando su clase y quizás un valor inicial por defecto. Normalmente se pone en mayúsculas la primera letra de cada palabra, excepto la primera, por ejemplo:

- nombre
- esMaestra

Operación. Implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre un comportamiento.

Firma. Una operación se puede especificar indicando su firma, la cual incluye:
- Nombre
- Tipo de parámetros
- Valores por defecto de todos los parámetros y un Tipo de retorno de la operación

Protocolo. Es el conjunto de firmas de todas las operaciones de un objeto.

Estereotipos
- Un estereotipo es un elemento que extiende las características
- Cada clase puede tener como máximo un estereotipo
- Estereotipos comunes.
  - Entity.
  - Boundary (interfaz).
  - Control
- Los Estereotipos se muestran en la parte donde se escribe el nombre de la clase entre << >>

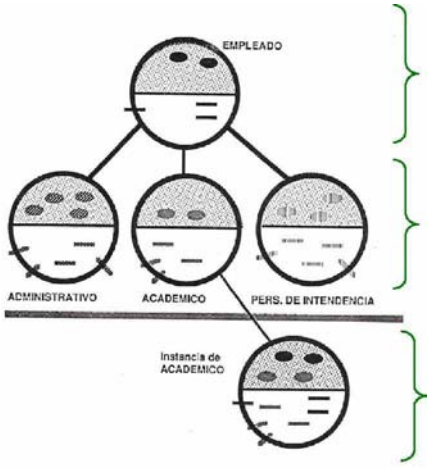
Clases Entity
- Una clase entity modela información y comportamiento asociado que es generalmente de larga vida (persistente).
- Son de alto nivel (perspectiva conceptual) también puede necesitarse para las tareas internas del sistema ya que corresponden a las estructuras de datos (mapeo a BD).
- Representan los objetos de negocio (análisis). Ejemplos de clases entity:
  - Producto
  - Cliente

Clases Boundary
- Las clases boundary se utilizan para modelar la interacción entre el sistema y sus actores. Esta interacción a menudo implica recibir (y presentar) información y peticiones de (y hacia) los usuarios y los sistemas externos.
- Representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de comunicaciones, impresoras, sensores, terminales. Ejemplos:
  - frmInscripcion
  - formaDatosPersonales

Clases Control
- Las clases de control representan coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto.
- Las clases de control también se utilizan para representar derivaciones y cálculos completos, como lógica del negocio que no pueden asociarse con ninguna información concreta, de larga duración, almacenada por el sistema.

Herencia
- Relación para reusar clases existentes y así definir nuevas. Las clases hijas conservan la estructura y comportamiento de las clases padre.
- La clase general es llamada superclase; la especialización subclase.
- La superclase define un comportamiento (protocolo) que todas las subclases heredan
- Las subclases pueden agregar nuevas atributos y operaciones
- Las subclases pueden sobre-escribir operaciones (conservando la firma pero proporcionando distintas implementaciones)

Polimorfismo
Es la propiedad por la cual dos o más clases responden al mismo mensaje cada uno de forma diferente. Realizan la misma operación de forma diferente para cada objeto.



Jerarquía

Un sistema se compone de subsistemas (más pequeños) relacionados que tienen a su vez propios subsistemas y así sucesivamente.

- Clasificación u ordenación de abstracciones.
  - De Tipos (Generalización/Especialización)
    - "Es un"
  - De Partes (Agregación)
    - "Es parte de"

Modularidad

- La modularidad es la propiedad que permite subdividir una aplicación en partes más pequeñas llamada módulos. Cada uno de estos módulos puede ser tan independientes como sea posible de la aplicación y de las otras partes.

Persistencia

- Conservación de la información en algún medio con la posibilidad de recuperarla en cualquier momento.
- Permite que los objetos existan fuera del programa en ejecución.

Mensaje

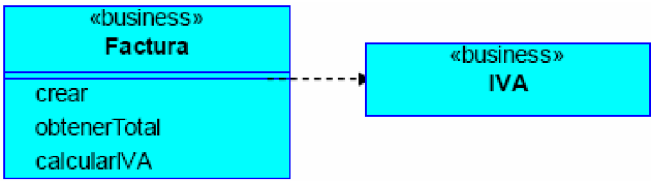
- Un mensaje es la petición de un servicio. Para que los objetos de un sistema trabajen en conjunto, un objeto (cliente) envía a otro una llamada para realizar una operación y el objeto receptor (proveedor) ejecutará la operación.
- El mensaje posee un emisor, un receptor y una acción.
  - Un mensaje es la petición de un servicio
  - La invocación de una operación es el tipo de mensaje más común
- Un mensaje es una función de llamada (con nombre, parámetros y un resultado)

Relaciones

- Todos los sistemas contienen varias clases y objetos
- Reflejan la colaboración entre clases.
- Los objetos contribuyen al comportamiento del sistema colaborando con unos y otros
- La existencia de una relación entre dos clases denota una comunicación (enlace) entre instancias de las clases, por las que un objeto puede enviar mensajes a otro
- Asociación
  - Nombre, rol, multiplicidad
- Agregación (pertenencia)
  - Relación de tipo tiene-un
- Generalización (herencia)
  - Relación de tipo es-un
- Dependencia

Dependencia

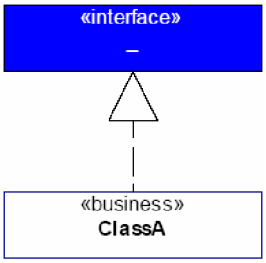
- Es una relación que declara que un cambio en la especificación de un elemento puede afectar a otro elemento que la utiliza, pero no necesariamente a la inversa.
- Gráficamente, una dependencia se representa como una línea discontinua dirigida hacia el elemento del cual se depende.



- Esta es una relación de uso, si la clase utilizada cambia, la operación de la otra clase puede verse también afectada, porque la clase utilizada puede presentar ahora una interfaz o comportamiento diferentes.
- También se utilizan en el contexto de las clases para indicar que una clase utiliza a otra como argumento en la signatura de una operación.
- Una dependencia puede tener un nombre, aunque es raro que se necesiten los nombres, a menos que se tenga un modelo con muchas dependencias.

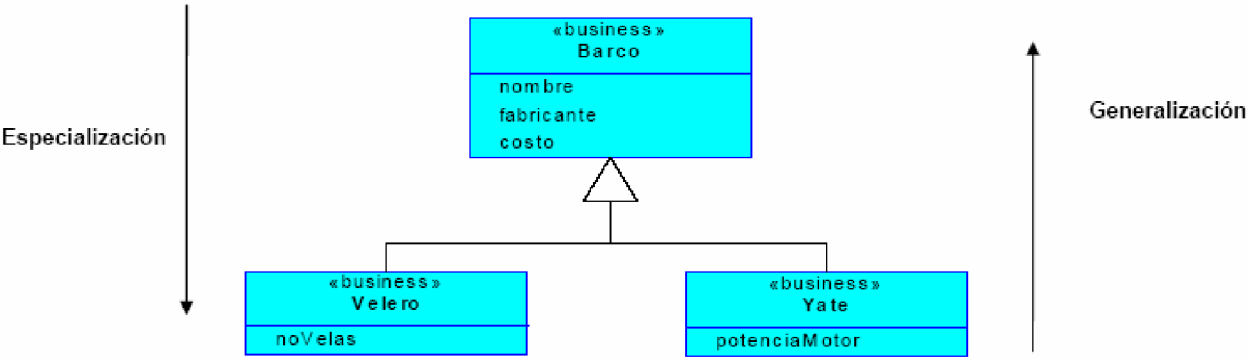
Realización

- Es una relación entre una especificación y su implementación.
- Es una relación semántica entre elementos, en donde un elemento especifica un contrato que otro elemento garantiza que cumplirá. Se pueden encontrar entre:
  - Interfaces y clases
  - Casos de Uso



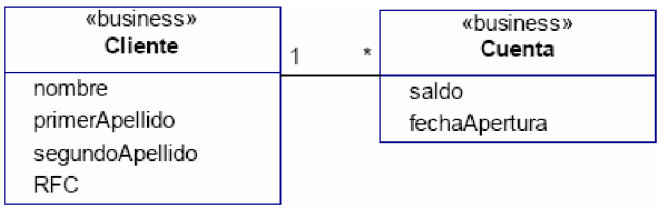
Generalización

- La generalización significa que los objetos hijos se pueden emplear en cualquier lugar que pueda aparecer el padre, pero no a la inversa.
- Una operación de un hijo con la misma signatura que una operación del padre pero que redefine la operación del padre se conoce como polimorfismo.
- La generalización permite conectar clases generales con otras más especializadas
- Se conocen como subclase/superclase o hijo/padre
- Es un relación de “es-un-tipo-de”
- Una clase hija hereda las propiedades de sus clases padres, especialmente sus atributos y operaciones



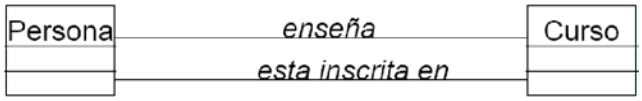
Asociación

- Una asociación es una conexión semántica bi-direccional entre clases
  - Esto implica que hay una liga entre objetos en las clases asociadas
- Dada una asociación entre dos clases, se puede navegar desde un objeto de una clase hasta un objeto de la otra clase y viceversa a través de la liga.
- Ambos extremos de una asociación pueden estar conectados a la misma clase.
- Gráficamente una asociación se representa como una línea continua que conecta la misma o diferentes clases.



Nombre de una asociación

Ø Se utiliza para describir y clarificar la naturaleza de la relación.
Ø Para que no haya ambigüedad en su significado, se puede dar una dirección al nombre a lo largo de la línea por medio de una flecha que apunte a la dirección en la que se pretende que se lea el nombre. Si se tiene un modelo con muchas asociaciones o para cuando se tiene más de una asociación entre las mismas clases.
Ø Un nombre de asociación es usualmente un verbo o una frase con verbo



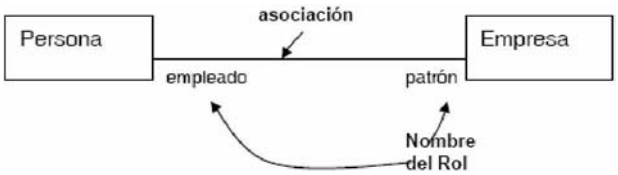
Relación "Uses o Extends"

La relación Uses o Include ocurre cuando se tiene una porción de comportamiento que es similar en más de un caso y no se quiere duplicar la descripción de tal conducta.

Ø Uses o include permite incluir la misma funcionalidad en dos o más Caso de Uso separados sin necesidad de repetir los detalles.
Ø Se utiliza la relación extend cuando se tiene un Caso de Uso que es similar a otro, pero que hace un poco más. Extend describe una variación de la conducta normal.

ROL

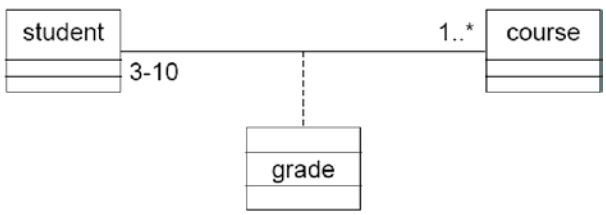
Ø Es simplemente la cara que la clase de un extremo de la asociación presenta a la clase del otro extremo.
Ø Un rol denota el propósito o capacidad en la que una clase se asocia con otra
Ø Los nombres de roles son típicamente sustantivos o frases con sustantivo
Ø Un nombre de rol se pone a lo largo de la línea de asociación cerca de la clase que modifica
Ø Uno o ambos finales de una asociación pueden tener nombres de roles



Clase asociación

Ø Si quisiéramos rastrear los grados para todos los cursos que un alumno ha tomado
  Ø La relación entre alumno y course es una relación de muchos-a-muchos

  Ø Crear una clase de asociación usando el icono clase
  Ø Conectar el icono clase a la línea de asociación usando una línea punteada
  Ø La clase de asociación puede incluir múltiples propiedades de la asociación
  Ø Sólo se permite una clase de asociación por asociación



Multiplicidad

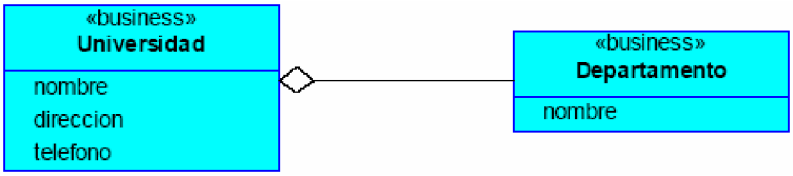
Ø Se usa para señalar cuántos objetos pueden conectarse a través de una instancia de una asociación.

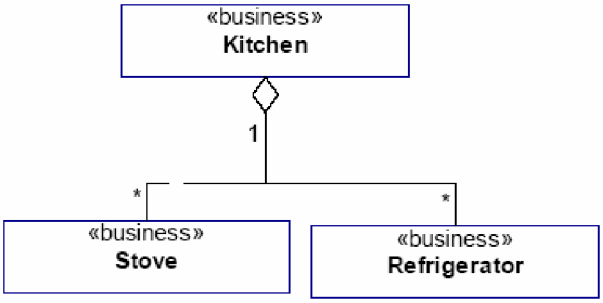
Ø Para cada asociación, hay dos decisiones de multiplicidad que tomar: una por cada final de la asociación

Muchos	*
Exactamente uno	1
Cero o más	0.*
Uno o más	1.*
Cero o uno	0.1
Rango especifico	2 4

Agregación

Ø La agregación es una forma especializada de asociación en la que un todo se relaciona con su parte o sus partes
  Ø Agregación es conocida como "parte de" o relación que contiene
Ø Una agregación se representa como una asociación con un diamante al lado de la clase denotando el agregado (todo)
Ø Se utiliza para modelar una relación "todo/parte".

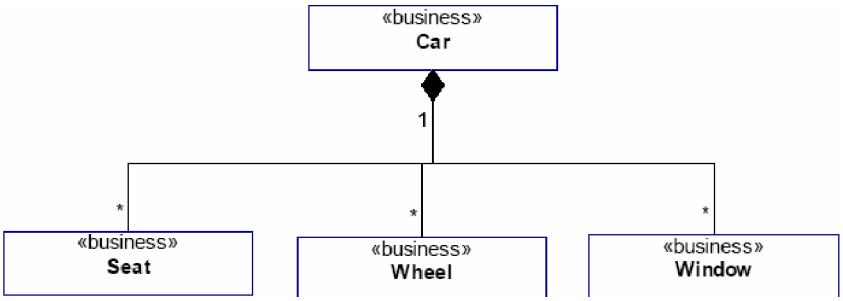




Composición

Es una especialización de la agregación en la que las partes no pueden existir independientemente del objeto completo.

  Ø Por ejemplo: Un humano es una composición de una cabeza, dos brazos, dos piernas y un torso. Si se quita alguna de las partes sin intervención quirúrgica el persona entera morirá

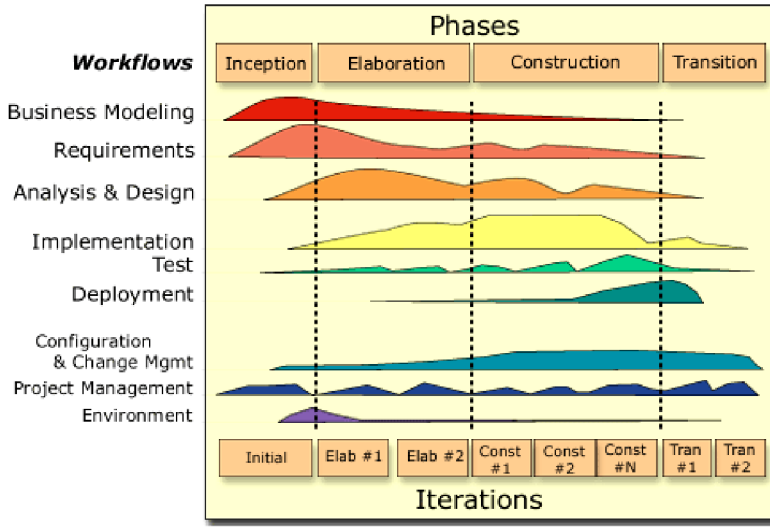


Qué es un proceso de desarrollo

Ø Un proceso es un conjunto de actividades, métodos y prácticas para desarrollar y mantener el software y los productos asociados
Ø Un proceso define qué hacer, cuándo y cómo hacerlo y quién(es) deben hacerlo.

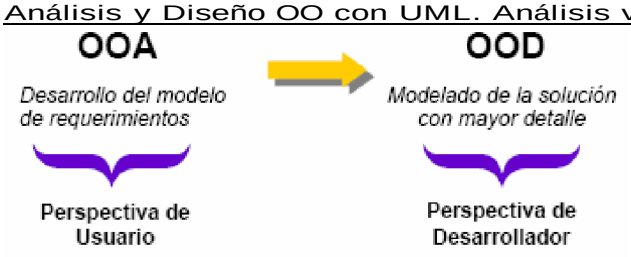
Proceso de Desarrollo Iterativo e Incremental RUP

El Proceso Unificado de Rational (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización



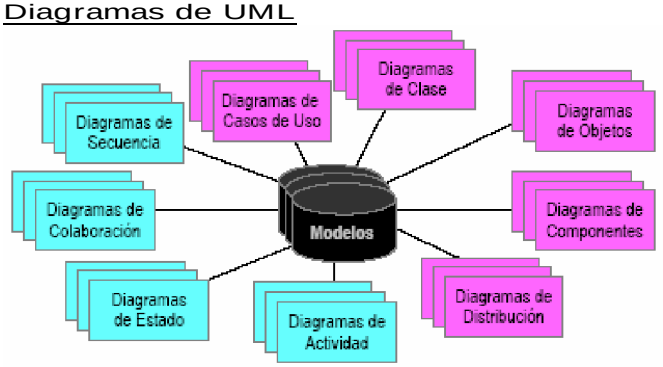
Análisis y Diseño Orientado a Objetos con UML

Análisis y Diseño OO con UML. Análisis vs. Diseño



- Análisis
  - Determinar el qué se va a hacer
  - Obtener, analizar y especificar requerimientos
  - Modelar sistema desde la perspectiva del usuario
  - "Problema"
- Diseño
  - Determinar el cómo se va a hacer (conceptualmente)
  - Realizar especificaciones de software
  - Modelar solución desde la perspectiva del desarrollador
  - "Solución"

Diagramas de UML



Perspectivas de una Aplicación

- Casos de Uso (Requerimientos)
  - Diagramas de Casos de Usos

- Ø Diagrama de Actividades
- Ø Diagrama de Secuencia
- Ø Lógica (Estructura Interna)
  - Ø Diagrama de Clases
  - Ø Diagrama de Interacción
  - Ø Secuencia/Colaboración
  - Ø Diagrama de Estados
- Ø Componentes (artefactos)
  - Ø Diagrama de Componentes
- Ø Distribución (Distribución Física)
  - Ø Diagrama de Distribución

Requerimientos

¿Qué es un requerimiento?

- Ø De manera general, un requerimiento es una condición o característica que debe satisfacerse.
- Ø En el contexto de desarrollo de software, un requerimiento es una característica identificable expresada en términos de funcionalidad o desempeño que un sistema debe poseer para lograr su objetivo.

Niveles de requerimientos

Dominio del Problema

Dominio del la Solucion

De negocio (necesidades)

- Ø Representan las necesidades de los usuarios y otros involucrados. Puede estar asociado a un problema operacional o una oportunidad del negocio. Ejemplos:
  - Ø Contar con información oportuna de los empleados para la toma de decisiones.
  - Ø Facilitar el cálculo de la nómina.

De sistema (características)

- Ø Descripciones simples en el lenguaje del usuario que se utilizan para comunicar la solución de alto nivel a los involucrados. Son los servicios que el sistema proporciona para satisfacer las necesidades. Ejemplos:
  - Ø El sistema registrará los datos personales de los empleados.
  - Ø El sistema permitirá el almacenamiento de la fotografía de los empleados.

De software (especificaciones)

- Ø Describen de manera más específica la solución. Canalizan las características en soluciones de software específicas. Ejemplos:
  - Ø Creación de un componente (Web Services) para el cálculo de nómina.
  - Ø Generación de una consulta SQL parametrizable para la clase de empleado.

Tipos de requerimientos

Requerimientos funcionales.

- Ø Describen lo que el sistema debe hacer, es decir especifican acciones que el sistema debe ser capaz de realizar (funcionalidad).

Requerimientos no funcionales

- Ø Describen atributos del sistema o atributos del ambiente del sistema, como son: facilidad de uso, confiabilidad, desempeño, de diseño, de interfaz, legales, de seguridad.

Casos de Uso

- Ø En conjunto representan toda la funcionalidad del sistema
- Ø Describen al sistema, su ambiente y las relaciones entre estos
- Ø Nos permiten modelar y especificar los requerimientos funcionales de un sistema.
- Ø Son una toma instantánea de algún aspecto del sistema
- Ø Especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo y que producen un resultado observable de valor para un actor
- Ø Es una interacción típica entre un usuario (Actor) y un sistema
- Ø Es uno de los servicios que ofrece un sistema a un Actor.
- Ø Es una unidad coherente de funcionalidad
- Ø Fue un concepto aportado por Jacobson en 1994
- Ø Modela las posibles formas en que un sistema puede ser usado
- Ø Especifican el comportamiento del sistema y sus requerimientos
- Ø La Vista de Casos de Uso captura el cómo actúa y reacciona el sistema: la actividad visible y comprobable comportamiento de un sistema

Casos de Uso-Fuentes

- Ø Conversación con el cliente/usuario
- Ø Dominio del problema
- Ø Especificación del sistema
- Ø Literatura relevante del dominio
- Ø Entrevistas con expertos
- Ø Conocimiento personal del dominio
- Ø Sistemas legados
- Ø Experiencia de sistemas anteriores

Utilidad de los Casos de Uso

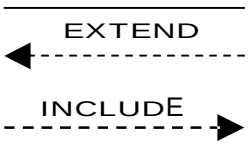
- Ø Clientes. Aprueban lo que el sistema debe hacer
- Ø Usuarios. Ganan entendimiento del sistema
- Ø Analistas. Proporciona las bases para el análisis y diseño
- Ø Desarrolladores. Comportamiento del documento del sistema
- Ø Probadores. Se usan como base para las pruebas del sistema.
- Ø Líder del Proyecto. Planeación de proyectos
- Ø Documentador. Base para la guía de usuario

Casos de Uso –Notación

Un Caso de Uso es una interacción típica entre un usuario (Actor) y un sistema.

ACTOR

Un Actor representa cualquier cosa que interactúe con el sistema.



Las asociaciones y dependencias permiten representar interacciones entre un actor y un caso de uso o entre casos de uso.

Actores

- Ø Los Actores no son parte del sistema; representan los roles que pueden jugar los usuarios del sistema.
- Ø Un Actor puede intercambiar información activamente con el sistema (asociación bidireccional).
- Ø Un Actor puede ser un receptor pasivo de información.
- Ø Un Actor puede ser una persona, máquina u otro sistema que interactua con el sistema.
- Ø Cada Actor participa en uno o más Casos de Uso.
- Ø Un Actor puede ser representado por distintas personas.
- Ø Una persona puede desempeñar varios roles.

Actores-Cómo encontrarlos

- Ø ¿Quién está interesado en cierto requerimiento? ¿Qué papel desempeña?

- Ø ¿Dónde se usara el sistema dentro de la organización? ¿En qué áreas?
- Ø ¿Quién proveerá de información al sistema? ¿Quién consultará la información? ¿Quién borrará la información?
- Ø ¿Quién usará determinada función?
- Ø ¿Quién le dará soporte y mantenimiento al sistema?
- Ø ¿El sistema usa una fuente externa?
- Ø ¿Qué actores necesitan los casos de uso?
- Ø ¿Puede un actor desempeñar varios roles diferentes?
- Ø ¿Varios actores desempeñan el mismo rol?

Especificación de Casos de Uso

Caso de Uso	Nombre del Caso de Uso				
Id del CU	#	Prioridad	Alta Media Baja	Estado	En Elaboración Propuesto Validado
Actores	Actor o Actores relacionados con el Caso de Uso				
Breve Descripción	Se explica en pocas líneas el caso				
PRE – Condiciones	Requisitos necesarios para que el Caso de Uso se lleve a cabo Del Proceso Relacionados con el proceso de negocio, independientes del sistema Del Sistema Directamente relacionados con el sistema				
Flujo Principal	Secuencia de acciones que se dan normalmente				
Flujos Alternos	Secuencia de acciones que detallan las alternativas del caso				
Flujos de Excepción	Eventos extraordinarios del Caso				
Post – Condiciones	Requisitos que deben cumplirse posteriormente				

Documentación de los Casos de Uso

- Ø Describe sólo los eventos que pertenecen al caso de uso y no lo que pasa en otros casos de uso
- Ø Utilizar un lenguaje comprensible para el cliente
- Ø Evitar terminología vaga como “por ejemplo”, “etc.”, “los datos”, etc.
- Ø Deberá describir:
  - Ø Cómo y cuándo inicia y termina el caso de uso
  - Ø Cuando interactúa el caso de uso con los actores
  - Ø Qué información se intercambia entre un actor y el caso de uso
  - Ø No describe los detalles de la interfaz de usuario, describe las acciones

Narrativa simple

El usuario indica que desea “Recuperar contraseña”. El sistema despliega la pregunta secreta elegida por el usuario al momento de registrarse y el usuario ingresa la respuesta. El sistema verifica que la respuesta sea coincidente y envía la nueva contraseña al correo electrónico del usuario. La contraseña es generada automáticamente por el sistema de forma aleatoria, la cual está formada como sigue:

2 números + 2 letras mayúsculas + 2 números + 1 carácter especial

Secuencia

1. El usuario indica que desea “Recuperar contraseña”.
2. El sistema despliega la pregunta secreta elegida por el usuario al momento de registrarse.
3. El usuario ingresa la respuesta.
4. El sistema verifica que la respuesta sea coincidente.
5. Si es coincidente, el sistema envía la nueva contraseña al correo electrónico del usuario, la cual está formada por:

2 números + 2 letras mayúsculas + 2 números + 1 carácter especial

Especificación caso de uso: Conversación

Acciones de Usuario	Responsabilidades del Sistema
Indica Recuperar Contraseña	Despliega la pregunta secreta elegida por el usuario
Ingresa la respuesta	Verifica que la respuesta sea coincidente Si es coincidente el sistema envía la nueva contraseña al correo electrónico del usuario la cual esta formada por:

	2 números + 2 letras mayúsculas + 2 números + 1 carácter especial

Consideraciones
- Los casos de uso deben escribirse utilizando el lenguaje del cliente.
- El nombre debe denotar una acción (verbo), como infinitivo o como sustantivo, por ejemplo: Generar informes ó Generación de informes.
- Redactar en presente y en voz activa (no pasiva).
- Cada caso de uso debe estructurarse para que forme una especificación de funcionalidad completa e intuitiva. No deberían estructurarse sólo casos de uso pequeños y no intuitivos para reducir redundancias; debe llegarse a un equilibrio entre comprensibilidad y mantenibilidad.
- Describir claramente la información que se intercambia entre actor y caso de uso

Las especificaciones de requerimientos no deben considerar aspectos propios de la implementación ni tecnicismos. Sin embargo los diseñadores y probadores necesitan un mayor detalle por lo que es altamente recomendable enfatizar lo siguiente:
- Entradas y salidas de información
- Restricciones y validaciones
- Detalle de los algoritmos
- Características de los datos (obligatoriedad, longitud, tipo)
- También pueden incluirse "Notas técnicas" dentro del Caso de Uso, identificadas de tal manera que no se confundan con la especificación base ya que son irrelevantes para el usuario.
- Cada Caso de Uso representa una forma de usar el sistema (dan soporte a un usuario durante un proceso de negocio)
- Los mejores Caso de Uso son aquellos que añaden el mayor valor al negocio que implanta el sistema
  - Casos de Uso vs. "Casos de Abuso"
  - Casos de Uso vs. "Casos sin Uso"
- Es necesario ajustar la granularidad de acuerdo al problema:
  - Muchos Casos de Uso pueden ser abrumantes
  - Pocos Casos de Uso pueden ocultar detalles importante

Escenarios
- Un escenario es una instancia de un caso de uso.
- Es una ruta que muestra una combinación particular de condiciones en un caso de uso.
- Un Caso de Prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para probar un camino concreto a través de un Caso de Uso
  - Escenarios primarios. Flujo normal. La forma en la que debe trabajar normalmente.
  - Escenarios secundarios. Excepciones del escenario primario

Diagramas complementarios
- Pueden utilizarse los diagramas de estado para describirlos estados de algún objeto de negocio derivado de los casos de uso y las transiciones entre estos.
- Pueden utilizarse los diagramas de actividad para describirlas secuencias de acción de un caso de uso, los flujos de trabajo asociados a un caso de uso o la secuencia de varios casos de uso del sistema.
- Pueden utilizarse los diagramas de secuencia para describir la interacción entre el actor y el caso de uso.

Diagramas

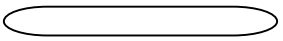
Diagramas de Actividad
- Los diagramas de actividades muestran como la secuencia de actividades, acciones paralelas y las dependencias entre ellas.
- Son útiles para la descripción del comportamiento y procesos en paralelo.
- Permiten modelar flujos de trabajo y Casos de Uso.
- Los diagramas de actividades pueden dividirse en "carriles" (swimlanes) que determinan qué objeto es responsable de qué actividad, muestran además de qué pasa, quién lo hace.
- Desde la perspectiva de la implementación, permite representar qué clase es responsable de qué actividad, mientras que desde el punto de vista de dominio, permite presentar qué persona o departamento es responsable de qué actividad.
- El diagrama de actividad combina ideas de varias técnicas: el diagrama de eventos de John Odell, las técnicas de modelado de estados de SDL y las redes de Petri

Ø Permite modelar los procesos reales de una organización humana y modelar actividades de software

Notación

Inicio

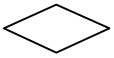
Fin



Actividad. Es una acción a realizar

Barra de sincronización. Permite iniciar acciones una vez que se han realizado actividades concurrentes o bien marcar el inicio de varias actividades en paralelo.

Carriles. Líneas verticales que permiten dividir la responsabilidad de las actividades.

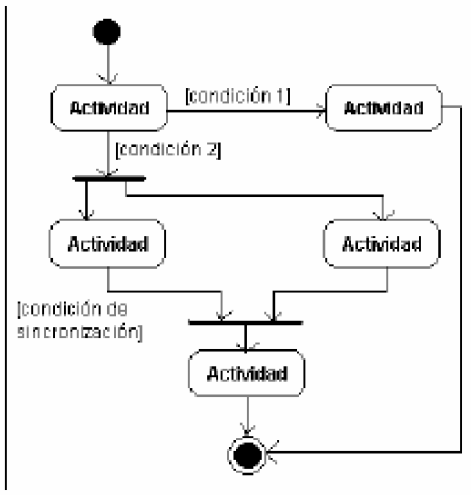


Decisión. Es un punto en el que se pueden seguir alternativas distintas de acuerdo al resultado de la actividad anterior

Transición. Indica la secuencia de las actividades.

Las condiciones de guarda son los posibles resultados de una acción que servirán como condición para la realización de otra

Diagramas de Actividad



Swimlanes

Ø Los diagramas de actividad mediante los “swimlanes”muestra además de qué pasa, quién lo hace
  Ø Se indican con una línea vertical que separa el diagrama en zonas. Cada zona representa una clase, persona o departamento en particular
Ø Utilidad:
  Ø Desde la perspectiva de la implementación, permite representar qué clase es responsable de qué actividad
  Ø Desde el punto de vista de dominio, permite representar qué persona o departamento es responsable de qué actividad

AND/OR

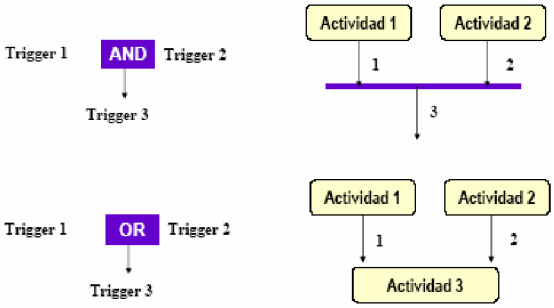


Diagrama de Clases

- Se usa para modelar la vista de diseño estática de un sistema. Soporta principalmente requisitos funcionales de un sistema, es decir, los servicios que el sistema debe proporcionar a sus usuarios finales. Representan las características estructurales
- Muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones estáticas (asociación y agregación) entre ellas
- Los diagramas de clases contienen normalmente los siguientes elementos:
  - Clases
  - Interfaces
  - Colaboraciones
  - Relaciones de dependencia, generalización y asociación.
  - Notas y restricciones
- También pueden contener paquetes o subsistemas, los cuales se usan para agrupar los elementos de un modelo en partes más grandes
- Se consideran la columna vertebral de los métodos OO

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 diagrama de clases bien estructurado:

- Se centra en comunicar un aspecto de la vista de diseño estática de un sistema.
- Contiene sólo elementos que son esenciales para comprender ese aspecto.
- Proporciona detalles de forma consistente con el nivel de abstracción, mostrando sólo aquellos adornos que sean esenciales para su comprensión.

Cuando se dibuje un diagrama de clases:

- Darle un nombre que comunique su propósito.
- Distribuir sus elementos para minimizar los cruces de líneas.
- Organizar sus elementos espacialmente de modo que los que estén cercanos semánticamente también lo estén físicamente.
- Usar notas y colores como señales visuales para llamar la atención sobre características importantes del diagrama.

Diagramas de Interacción

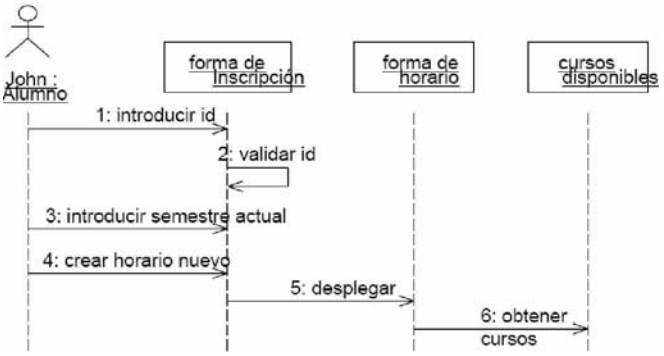
Una interacción establece el escenario para un comportamiento del sistema introduciendo todos los objetos que colaboran para realizar alguna acción. Incluyen los mensajes enviados entre objetos. La mayoría de las veces, un mensaje implica la invocación de una operación o el envío de una señal.

Las interacciones se usan para modelar el flujo de control dentro de una operación, una clase, un componente, un caso de uso o el propio sistema.

- Un diagrama de interacción es una representación gráfica de interacciones entre objetos. Hay dos tipos de diagramas de interacción:
  - Diagramas de secuencia
  - Diagramas de colaboración
- Cada uno provee un punto de vista diferente de la misma interacción
  - Los diagramas de secuencia están ordenados de acuerdo al tiempo
  - Los diagramas de colaboración muestran la organización entre objetos. Pueden incluir flujo de datos

Diagrama de Secuencia

- Un diagrama de secuencia muestra interacciones de objetos ordenados en secuencia de tiempo. El diagrama muestra:
  - Los objetos que participan en la interacción
  - La secuencia de mensajes intercambiados
  - Enfoque de control (opcional)

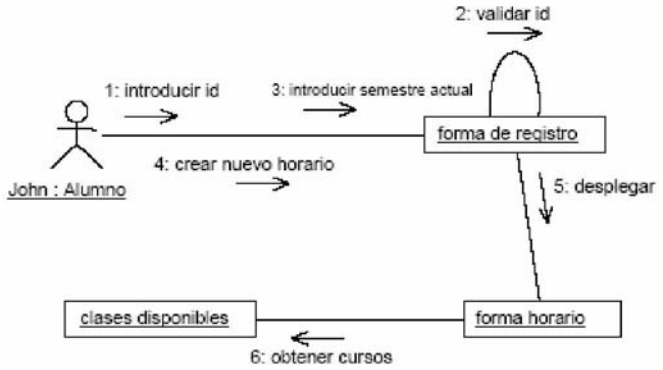


Scripts para Diagramas de Secuencia

- Para escenarios complejos, los diagramas de secuencia pueden ampliarse mediante el uso de scripts.
- Un script es escrito a la izquierda de un diagrama de secuencia.
- Los scripts pueden ser escritos en un lenguaje natural o pseudocódigo.

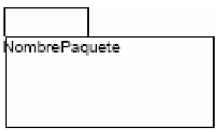
Diagramas de Colaboración

- Un diagrama de colaboración es una forma alternativa de representar los mensajes intercambiados por un conjunto de objetos.
- El diagrama muestra interacciones de objeto organizadas alrededor de los objetos y sus ligas a cada uno.
- Un diagrama de colaboración contiene:
  - Objetos
  - Ligas entre objetos
  - Mensajes intercambiados entre objetos
  - Flujo de datos entre objetos, si hay alguno



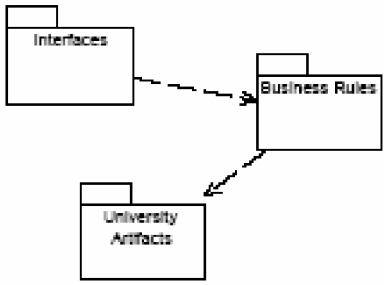
Paquetes

- Se utilizan para organizar elementos de modelado en partes mayores que se pueden manipular como un grupo.
- Un paquete es un mecanismo de propósito general para
  - organizar elementos en grupos
  - Se visualizan como carpetas
- Permiten controlar el acceso a sus contenidos para controlar las líneas de separación de la arquitectura del sistema.
- Los paquetes bien diseñados agrupan elementos cercanos semánticamente y que suelen cambiar juntos.



Relaciones entre paquetes

- Los paquetes se relacionan unos a otros usando una relación de dependencia. Existen dos tipos:
  - Dependencia. Si una clase en un paquete “habla” con una clase en otro paquete entonces se agrega una relación de dependencia en el nivel de paquete
  - Generalización.



Visibilidad
- Ø Public. Accesible a todas las clases
- Ø Protected. Accesible sólo a las subclases y a la clase misma
- Ø Private. Accesible sólo a la clase
- Ø Implementation. Accesible únicamente por la implementación del paquete

Evolución en Análisis y Diseño

Durante el análisis:
- Ø Se establecen clases (principalmente entity) y conexiones básicas (asociaciones, agregaciones y generalizaciones)
  - Ø Estas conexiones existen debido a la naturaleza de las clases y no debido a una implementación específica
- Ø Se hace una estimación inicial de multiplicidad para exponer hipótesis ocultas

Durante el diseño:
- Ø Se consideran otros estereotipos de clase (boundary y control)
- Ø Se refinan y actualizan las estimaciones de multiplicidad
- Ø Se evalúan y refinan las relaciones
- Ø Se detallan los tipos y firmas (signature)

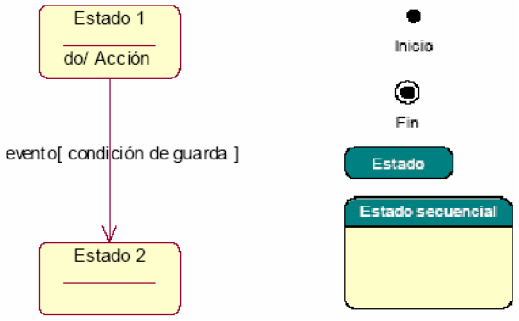
Diagramas de estado

- Ø Los diagramas de clase representan la estructura y el comportamiento estático de los objetos.
- Ø Los diagramas de estado se usan 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.
  - Ø El estado de un objeto es una de las condiciones posibles en las que puede existir.
- Ø Describe el comportamiento dinámico de los objetos en un cierto plazo.

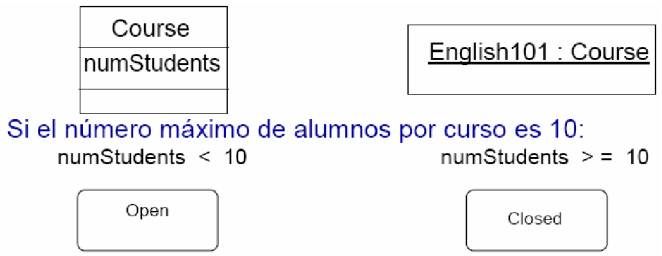
Notación

Un evento es la ocurrencia de alguna situación que sucede en un punto del tiempo y no tiene duración.
- Ø Evento de llamada.
- Ø Evento de señal.
- Ø Evento de cambio.
- Ø Eventos de tiempo

- Ø Una transición es un cambio de un estado original a un estado sucesor como resultado de algunos estímulos
- Ø Una acción es una operación que se asocia a una transición
- Ø Una actividad es una operación que toma tiempo para completarse. Las actividades se asocian con un estado
- Ø Una condición de guarda es una expresión booleana de valores de atributos que permiten una transición solo si la condición es verdadera



Ø Los estados pueden distinguirse por los valores de ciertos atributos



Ø Los estados también pueden identificarse por la existencia de ciertas ligas
  Ø Las instancias de la clase Profesor puede tener dos estados:
    Ø Impartir cuando existe una liga a un curso. En sabático cuando no existe liga

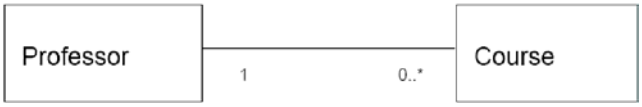


Diagrama de estado

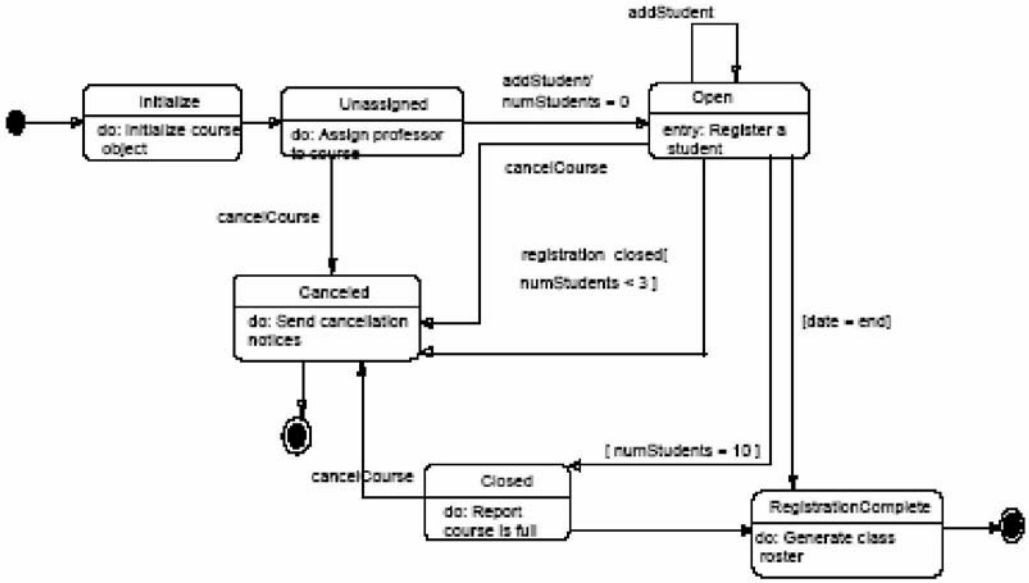
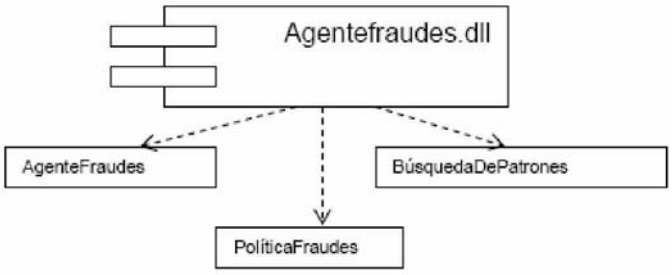


Diagrama de Componentes

Ø Muestran la organización y las dependencias entre un conjunto de componentes. Los diagramas de componentes se utilizan para modelar la vista física de implementación estática de un sistema. Esto implica modelar las cosas físicas que residen en un nodo, tales como ejecutables, bibliotecas, tablas, archivos y documentos.
Ø Un componente es una unidad de código fuente que sirve como bloque constructor para la estructura física de un sistema
Ø Los Componentes pertenecen al mundo real de los bits y, por tanto, son un bloque de construcción importante cuando se modelan los aspectos físicos de un sistema. El modelo físico se hace para construir el sistema ejecutable.
Ø Un componente es una parte física y reemplazable de un sistema conformado por un conjunto de interfaces y proporciona la realización de esas interfaces.
  Ø Ejemplos: Ejecutables EXE, bibliotecas DLL, programas principales, headers, módulos, formas, tablas, archivos y documentos
Ø Contenido de los diagramas de componentes:
  Ø Componentes.
  Ø Interfaces.
  Ø Relaciones de dependencia, generalización, asociación y realización.
  Ø Notas y restricciones
Ø Se pueden utilizar para:
  Ø Modelar código fuente.
  Ø Modelar versiones ejecutables.
  Ø Modelar bases de datos físicas.

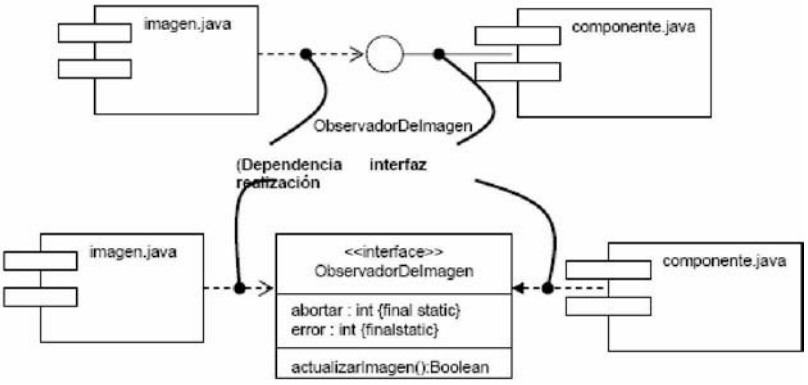
Componentes y clases

- Un componente es la implementación física de un conjunto de otros elementos lógicos, tales como clases y colaboraciones. La relación entre un componente y sus clases puede representarse explícitamente mediante una relación de dependencia.
- Tanto los componentes como las clases pueden realizar una interfaz, pero los servicios de un componente normalmente sólo están disponibles a través de sus interfaces.



Componentes e interfaces

- Una interfaz es una colección de operaciones que se utiliza para especificar un servicio de una clase o un componente.
- Para la implementación de componentes, primero se descompone la implementación física mediante la especificación de interfaces que representen las principales líneas de separación del sistema. A continuación, se proporcionan componentes que realicen estas interfaces, junto con otros componentes que accedan a los servicios a través de las interfaces.

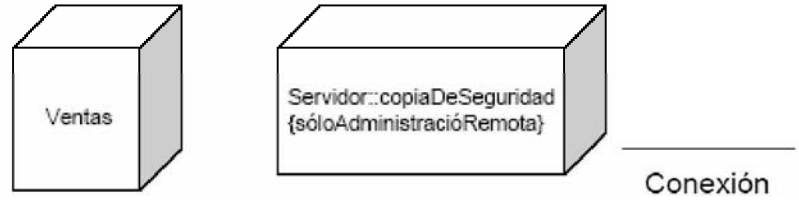


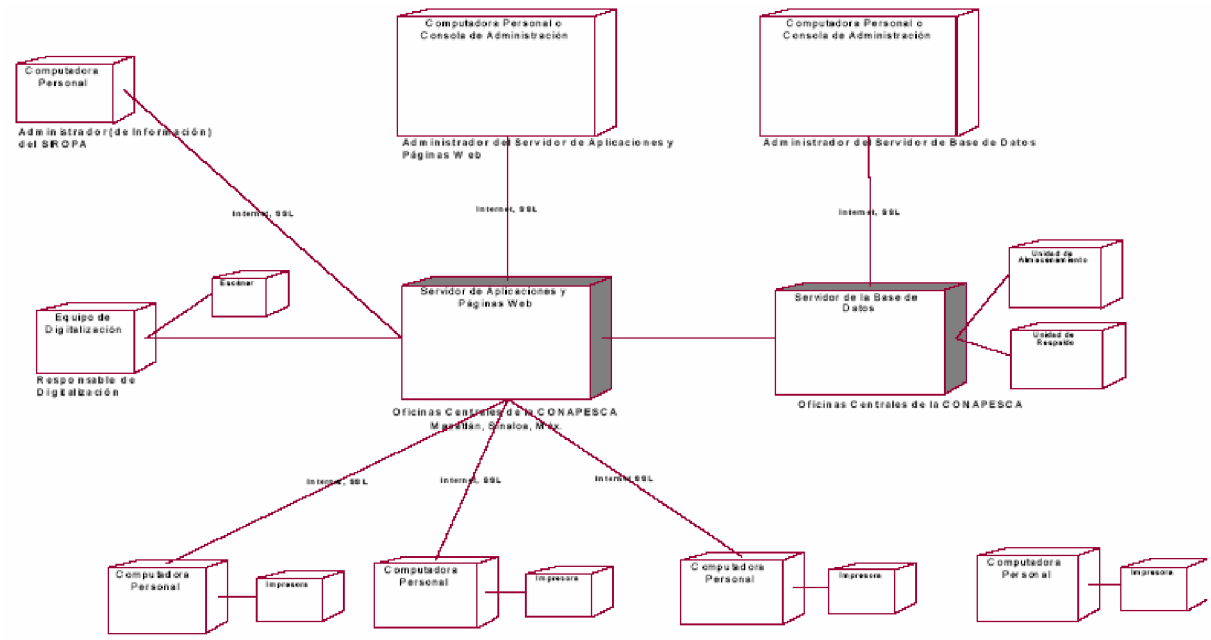
Tipos de Componentes

- Componentes de despliegue. Son los componentes necesarios para formar un sistema ejecutable, tales como las bibliotecas dinámicas (DLL's y EXE's).
- Componentes producto del trabajo. Son básicamente productos que quedan al final del proceso de desarrollo y consisten en cosas tales como archivos de código fuente y archivos de datos a partir de los cuales se crean los componentes de despliegue.
- Componentes de ejecución. Estos componentes se crean como consecuencia de un sistema en ejecución, tales como un objeto COM+, el cual se instancia a partir de una DLL.

Diagrama de Distribución

- Los diagramas de distribución son creados para mostrar los diferentes nodos (procesadores y dispositivos) en el sistema
- Los nodos, al igual que los componentes, pertenecen al mundo material y modelan el aspecto físico de un sistema
- Los nodos modelan la topología del hardware sobre el que se ejecuta el sistema
- Los elementos esenciales de un diagrama de distribución son los nodos y sus conexiones
  - Un nodo es un objeto físico que existe en tiempo de ejecución y representa un recurso computacional que tiene memoria y capacidad de procesamiento.
  - Una conexión indica comunicación, usualmente la relación directa entre hardware





Diagramas de Despliegue

Despliegue. Los componentes que se desarrollan o se reutilizan como parte de un sistema con gran cantidad de software deben desplegarse sobre algún hardware para su ejecución. Cuando se diseña un sistema con gran cantidad de software, hay que considerar tanto su dimensión lógica como la física:

- Ø En la parte física se encuentran los componentes (que representan los empaquetamientos físicos de los elementos lógicos) y los nodos (que representan el hardware sobre el que se despliegan y ejecutan esos componentes).

Diferencias entre los nodos y los componentes

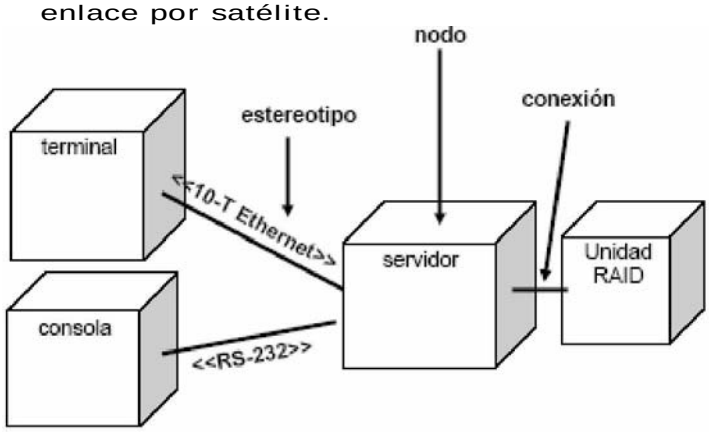
- Ø Los componentes son los elementos que participan en la ejecución de un sistema; los nodos son los elementos donde se ejecutan los componentes.
- Ø Los componentes representan el empaquetamiento físico de los elementos lógicos; los nodos representan el despliegue físico de componentes.

Atributos y operaciones en los nodos. Se pueden especificar atributos y operaciones para los nodos, por ejemplo, se podría especificar que un nodo tiene los atributos velocidadDelProcesador y memoria, así como las operaciones encendido, apagado y suspender.

Organización de Nodos. Los nodos se pueden organizar especificando relaciones de dependencia, generalización y asociación (incluyendo agregación) entre ellos.

Conexiones

- Ø El tipo más común de relación entre nodos es la asociación.
- Ø Una asociación representa una conexión física entre nodos, como puede ser una conexión Ethernet, una línea en serie o un bus compartido.
- Ø Se pueden utilizar asociaciones incluso para modelar conexiones indirectas, tales como un enlace por satélite.

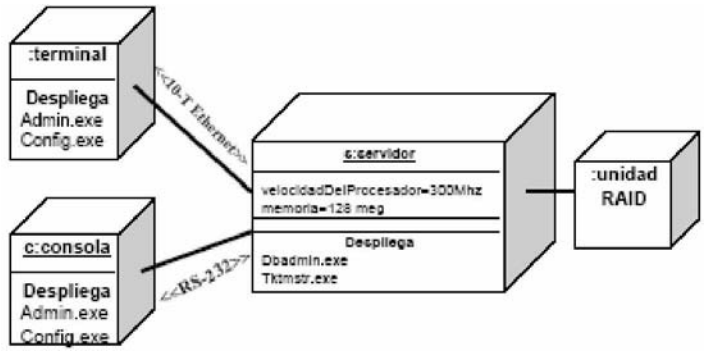


Modelado de la distribución de componentes

El siguiente diagrama especifica los componentes ejecutables que hay en cada nodo.

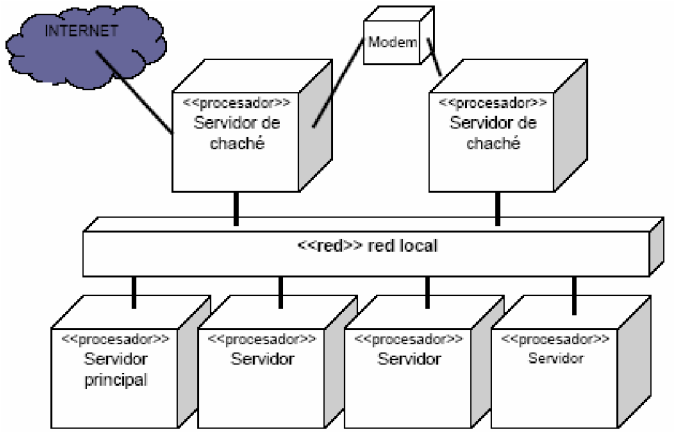
Cada procesador en la figura se representa con un compartimiento adicional que muestra los componentes que despliega.

El objeto servidor también se representa con sus atributos (velocidadDelProcesador y memoria) y lo valores de éstos.



Diagramas de despliegue

Se usan para modelar los aspectos físicos de los sistemas orientados a objetos. Muestra la configuración de nodos que participan en la ejecución y de los componentes que residen en ellos. Se utilizan para modelar la vista de despliegue estática de un sistema.



Ingeniería a partir del modelo de Objetos

Ingeniería a partir de Clases

Cómo modelar un esquema

- Identificación de clases persistentes (entity)
- Crear un diagrama de clases que contenga las clases marcadas como persistentes
- Expandir detalles estructurales de estas clases (tipos de datos, obligatoriedad)
- Centrar la atención en las relaciones que estructuran las clases y en sus cardinalidades
- Cada clase se mapea a una tabla

Perspectivas

4+1 Vistas

- Perspectivas diferentes para perfiles diferentes
  - Usuario final, cliente, administrador de proyecto
  - Ingeniero de sistema, desarrollador, arquitecto, evaluador
- Las perspectivas múltiples requieren múltiples vistas
  - Los diagramas de clase no muestran los mapas del sistema al hardware
  - Los diagramas de distribución no describen la estructura del sistema
- Para describir completamente una arquitectura, UML propone 4 vistas:
  - La vista lógica para proporcionar una imagen estática de las clases primarias y sus relaciones

- Ø La vista de componente para mostrar como está el código organizado en paquetes y librerías
- Ø La vista de proceso para mostrar los procesos y tareas
- Ø La vista de distribución para mostrar los procesadores, dispositivos y ligas en el ambiente operacional

Ø +1

- Ø Finalmente, una vista de escenario explica como trabajan juntas las otras cuatro vistas

Arquitectura

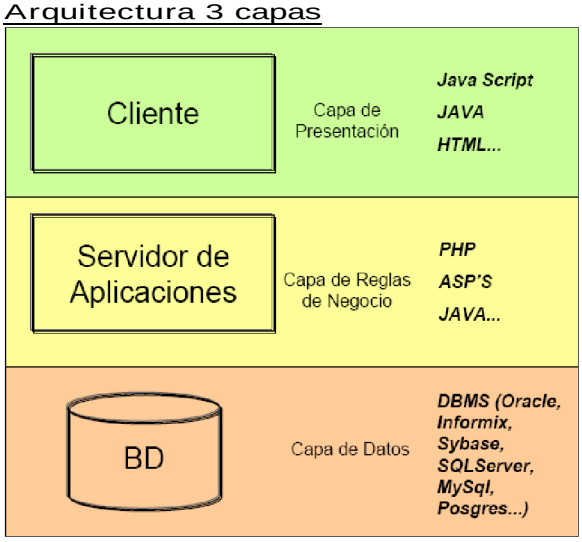
Arquitectura Multinivel

- Ø La arquitectura es la estructura organizativa de un sistema, que incluye su descomposición en partes, conectividad, mecanismos de interacción y principios de guía que proporcionan información sobre el diseño del mismo
- Ø La arquitectura es el espacio en donde trabajan los objetos
  - Ø ¿Cómo reparto mi sistema?
  - Ø ¿Qué herramientas se ajustan mejor?
- Ø Actualmente se definen dos estilos principales de arquitectura para las aplicaciones distribuidas:
  - Ø Arquitectura de 2 capas (Cliente/Servidor)
  - Ø Arquitectura de 3 capas (n capas)

Arquitectura Cliente/Servidor

- Ø Puede haber mucha carga en el cliente
- Ø Mucho tráfico en la red
- Ø Mantenimiento costoso, en cada cliente
- Ø Posibilidad de clientes desfasados
- Ø ¿Se usa una sola base de datos?
- Ø ¿La base de datos está situada en un único host?
- Ø ¿El tamaño de la BD será similar con el paso del tiempo?
- Ø ¿La cantidad de usuarios es similar a los largo del tiempo?
- Ø ¿Los requisitos son fijos o con muy poca posibilidad de cambiar?
- Ø ¿Se espera un mantenimiento mínimo de la aplicación?

Arquitectura 3 capas



Una arquitectura de tres capas:

- Ø Maximiza la reutilización de Objetos
- Ø Facilita el mantenimiento
- Ø Distribuye el procesamiento

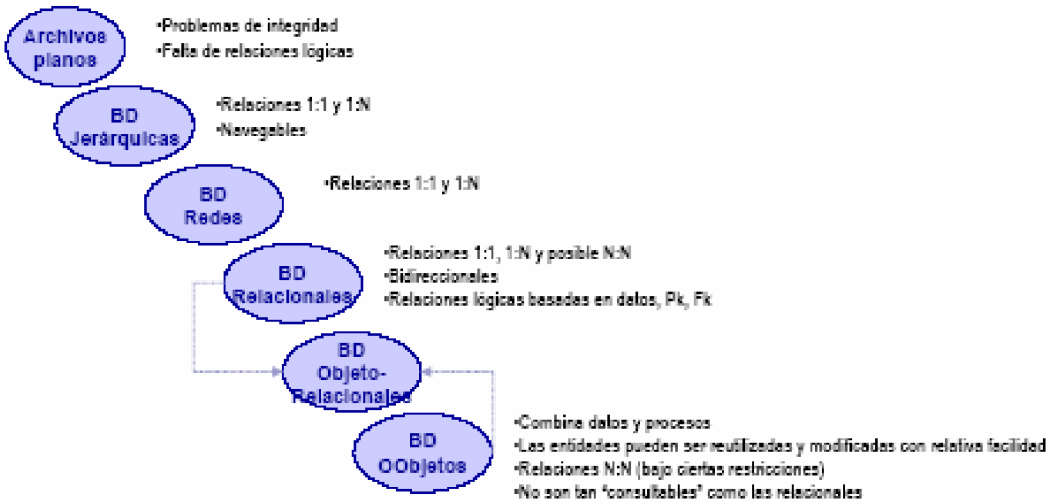
El principal inconveniente es el nivel de complejidad que da el sistema.

- Ø La capa de datos comprende lo relativo al almacenamiento lógico y físico de la información, es decir, las estructuras de datos así como los mecanismos de acceso, recuperación, procesamiento, seguridad y administración de la información (por ejemplo: base de datos relacional centralizada y gestionada por un Manejador de Bases de Datos X)
- Ø La capa de reglas de negocio tiene como objetivo especificar de manera formal los requerimientos funcionales propios del sistema, mediante la construcción de los componentes en alguna herramienta de desarrollo. Es intermediaria entre la capa de datos y la capa de presentación.
- Ø La capa de presentación es la que permite la interacción del usuario con el sistema; comprende por lo tanto las interfaces gráficas, la obtención y la presentación de información estática y dinámica al usuario, así como el procesamiento de la información a nivel del cliente, principalmente validaciones. Bajo esta arquitectura de tres capas en Web, si bien en la capa de presentación se realiza cierto procesamiento que permite agilizar y mejorar el desempeño del sistema, requiere recursos mínimos en la máquina cliente.

Bases de Datos Orientada a Objetos

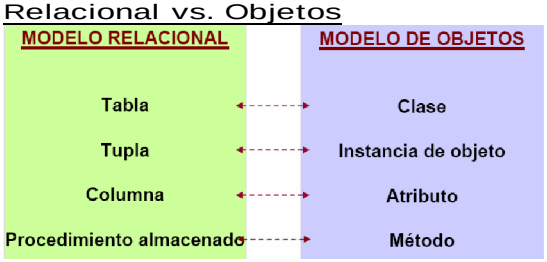
Introducción

La teoría del diseño de BD es una de las áreas de cómputo que más lentamente ha cambiado



El modelado de datos es el arte de identificar a partir de la realidad las entidades y sus relaciones que deben ser representadas en una base de datos.

Relacional vs. Objetos



Relacional	Objetos
Almacenan datos simples (caracteres, números)	Almacenan objetos
Representa tablas bidimensionales (columnas y renglones)	Representa clases (entidades con características y comportamiento)
Basado en la teoría de conjuntos y el álgebra relacional (unión, intersección, diferencia, Selección, proyección)	Basado en el paradigma orientado a objetos (clasificación, herencia, encapsulamiento)
Relaciones entre datos (FK's)	Relaciones entre objetos

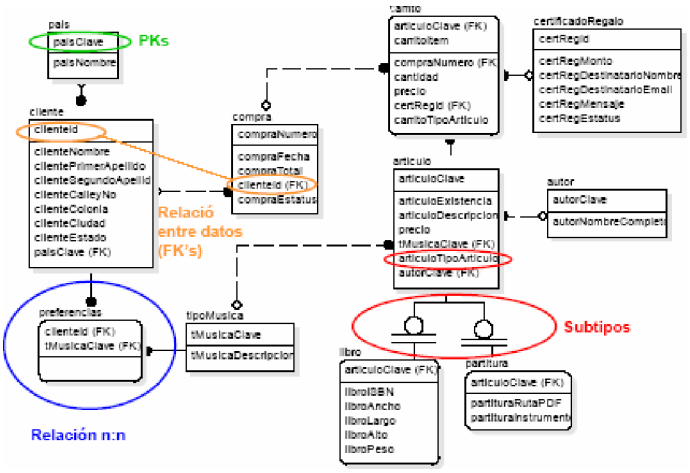
Objeto –Relacional
Es una base de datos relacional, pero soporta algunas características del OO

Modelo Relacional

Modelo Relacional

Bases de datos relacionales Las bases de datos relacionales almacenan datos (números enteros, cadenas, reales, etc.)

- Ø Almacenan la información en tablas bidimensionales (renglones y columnas).
- Ø Promueven la normalización.
- Ø Todas las columnas de la tabla dependen de una PK.
- Ø La información a almacenar se debe describir en términos de cadenas simples, enteros o números reales.



Modelo Relacional (Ventajas)

- Ø Ampliamente utilizadas y probadas
- Ø Madurez y diversidad del mercado
- Ø Modelado simple
- Ø Integración con prácticamente todas las herramientas de desarrollo.
- Ø Alta flexibilidad en la consulta y cruces de información
- Ø Transparencia
- Ø Cuentan con un lenguaje estándar: SQL, DDL, DML, DCL
- Ø Resuelven el 99.9999% de los casos

Modelado Orientado a Objetos

Bases de datos orientadas a objetos
- Almacenan objetos; no son renglones y columnas.
- Permiten manejar datos complejos.
- Soportan el paradigma OO
  - Identidad de los objetos
  - Tipos o clases
  - Datos (Estructura) + Procesos (Comportamiento)
  - Herencia - Jerarquía.
  - Encapsulamiento (Se complementan con los lenguajes de programación OO).

OODBs
- 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.
- Soportan toda la persistencia necesaria cuando se trabaja con lenguajes OO. Fueron diseñadas para ser integradas de forma transparente con la programación OO.
- Sistema administrador de bases de datos que soporta el modelado y creación de datos como objetos.

OO-DBMS
- Persistencia
  - Conservación de los datos; permanencia.
- Concurrencia
  - Interacción de muchos usuarios.
  - Recuperación
  - En caso de fallas de HW o SW, retroceder a un estado coherente de los datos.
- Gestión del almacenamiento secundario
  - Mecanismos no visibles al usuario para mantener la independencia lógica y física del sistema.
  - Facilidad de consultas

Objetos “Entity”. Los objetos “entity” usados por los programas OO son directamente análogos a las entidades de la base de datos OO.
- Los objetos definidos en un programa desaparecen una vez que el programa termina, es decir, son “transitorios”.
- Los objetos de una base de datos se conservan, es decir, son “persistentes”.

Características de interés para el modelado
- OID
  - Existencia de un identificador de objetos. No se requieren llaves primarias explícitas en todos los casos.
- Colecciones
  - Posibilidad de almacenar varios objetos y valores en un mismo atributo, mediante “SET” (conjuntos).
  - Navegabilidad
- Tipos personalizados (clases)
- Herencia

OIDs (Identificadores de Objeto)

Una base de datos relacional representa las relaciones mediante valores (PKs - FKs)	<==>	Una base de datos OO incluye identificadores de objetos dentro de un objeto indicando otros objetos al cual está relacionado

- Un OID es un identificador interno para cada objeto individual, los cuales nunca son manipulados por el “usuario”.
- El OID es independiente de cualquiera de sus características.
- Los OIDs son asignados y usados sólo por el DBMS.

Relaciones 1:N. En contraste con el modelo relacional, el OO permite almacenar valores múltiples. Esta característica es básica para cualquier tipo de relaciones “N”

Relaciones N:N. 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

Integridad de relaciones
- Se debe asegurar de alguna forma que los OIDs referenciados en un objeto coincidan con su contraparte

Cuando se inserta o se borra un OID, el OO-DBMS debe actualizar automáticamente la referencia a los objetos.

Navegabilidad. Se utiliza la navegabilidad en una relación no inversa entre dos objetos, donde solamente uno de los objetos contiene el OID del objeto relacionado.

Colecciones
- Las colecciones de objetos permiten a un objeto contener valores múltiples de una propiedad. El estándar del ODMG propone:
  - SET. Grupo no ordenado de objetos del mismo tipo. No acepta duplicados.
  - BAG. Grupo no ordenado de objetos del mismo tipo. Acepta duplicados.
  - LIST. Grupo ordenados de objetos del mismo tipo.
  - ARRAY. Grupo ordenado de objetos del mismo tipo que pueden ser accedidos por posición.
- Algunas de las operaciones que el ODMG establece para las colecciones:
  - Cardinality. Retorna el número de elementos en un colección
  - Is_empty. Retorna "true" si una colección está vacía
  - Is_ordered. Retorna "true" si los elementos están ordenados de alguna manera
  - Insert_element. Agrega un elemento a la colección
  - remove_element. Remueve un elemento

El que el OO-DBMS nos proporcione métodos para acceder a los conjuntos, promueve el encapsulamiento.

ODMG
- OMG – Object Management Group
  - ODMG – Object Data Management Group. ODMG 3.0
- ODL Object Definition Language. Es un lenguaje declarativo
- OQL Object Query Language. Es muy parecido al SQL-92. Con extensión para soportar conceptos de objetos
- OML (Object Manipulation Language)

Una de las razones del amplio uso de las bases de datos relaciones es la existencia de un lenguaje estándar de manipulación de datos (SQL)

ODL
- El ODL es usado para declarar la estructura de clases incluyendo propiedades y signature de las operaciones.
- La implementación de las operaciones requiere de un lenguaje de programación específico y por lo tanto no es parte del ODL.
- Intenta definir tipos que puedan implementarse en diversos lenguajes de programación, pero no está ligado a uno en particular.

```
CREATE TYPE POINT_TYPE AS OBJECT (x int y int)
CREATE TABLE CIRCLES ( RADIUS NUMBER, CENTER POINT_TYPE)
```

OQL
- Tiene una sintaxis abstracta
- Semántica formal
- Acceso declarativo a los objetos
- Se basa en el ODMG-93
- Tiene una sintaxis al estilo SQL
- Ejemplos:

```
Select x.age from Persons x where x.name = "Bob"
Select e.nombre from Empleado e where e.nombre = "Carlos"
Select struct(n: e.nombre, s: e.sexo) from Empleado as e where e.edad < 18
```

OML
- El lenguaje de manipulación es empleado para la elaboración de programas que permitan crear, modificar y borrar datos.
- ODMG-93 sugiere que este lenguaje sea la extensión de un lenguaje de programación, de forma que se puedan realizar por ejemplo, las siguientes operaciones:
  - Creación.
  - Borrado.
  - Modificación

- Ejemplos:
  ```
  void makePersistent (Object )
  void deletePersistent (Object )
  void makeTransient (Object ) ....
  ```

Resumen de beneficios

1. Modelado más apegado a la realidad.
   - Se identifican objetos, no renglones y columnas.
2. Transparencia en la programación OO.
   - No es necesario traducir de OO a SQL.
   - La manera en que se usan los datos en la programación es exactamente la misma en que se almacenan.
3. Desempeño en la administración de objetos complejos. No es necesario traducir de OO a SQL, ODBC o JDBC

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 son recomendadas cuando:
  - Se tienen pocos datos y éstos son muy simples (no son eficientes)
  - No se está usando un lenguaje OO
  - Se requiere una manipulación directa de los datos así como cruces de información.
  - La mayoría de las herramientas está orientada a RDBMS.
  - La mayoría de las empresas que las venden son relativamente pequeñas.

Aplicaciones de las BDOO

- CASE (Computer Aided Software Engineering)
- CAD (Computer-aided Design)
- CAM (Computer-aided Manufacturing)
- Telecomunicaciones
- Salud
- Finanzas
- Multimedia

Usos de las BDOO

- La bolsa de valores de Chicago administra el mercado de acciones vía ODBMS Versant.
- Radio Computing Services es la empresa de informática de radio más grande del mundo. Su producto, selector, automatiza todas las necesidades de la estación de radio, de la biblioteca de música, al salón de noticias, al departamento de ventas. RCS utiliza POETA ODBMS porque le permitió integrar y organizar varios elementos, sin importar tipos de datos, en un solo ambiente de programa
- La base de datos Objectivity/DB ODBMS se utiliza como depósito de datos para el nombramiento del componente de sistema, los datos para la planeación de misiones de satélites y los datos de administración orbital desplegado en el sistema Iridium.
- El centro médico de la universidad de Ajou en el Sur Corea utiliza InterSystems' Cachè ODBMS para apoyar todas funciones del hospital incluyendo departamentos de misión critica tales como patología, laboratorio, banco de sangre, farmacia, y radiografía
- El gran Collider Hadron en CERN en Suiza utiliza un Objectivity DB. La base de datos actualmente esta siendo probada en los centenares de Terabyte a velocidades de hasta 35 MB/segundo
- A noviembre del 2000, el centro acelerador linear de Stanford (SLAC) almacenó 169 terabytes de producción usando Objectivity/DB. Los datos de la producción se distribuyen a través de varios cientos de nodos de procesamiento y más de 30 servidores en línea.

OO-DBMS's

- Versant -Versant
- POET
- Objectivity DB -Objectivity
- ObjectStore – Object Design
- Jasmine – Computer Associates
- Gemstone – Servio Corp.
- Ontos –Ontos

Ejemplos:

```
CREATE TYPE direccion AS ( calleNo String; colonia String; ciudad String;
estado String; cp String;)
```

```
CREATE TYPE proveedor AS ( nombre String; dir direccion tel telefono www
String;)

select struct(n: p.name, s: p.sex) from People as p where p.age < 18
SELECT SName: p.name FROM p in People WHERE p.age > 26
SELECT s.name FROM Tutors t, t.students s

select c.address from Persons p, p.children c where p.address.street="Main
Street" and count(p.children) >= 2 and c.address.city != p.address.city

Constructors
Object: Employee(name: "John", salary: 15000)
```

Modelo Objeto Relacional

Objeto-Relacional. Son bases de datos relacionales pero soportan algunas características de la OO, por ejemplo

- Ø Tipos objetos
- Ø Colecciones
  - Ø VARRAYS
  - Ø Nested Tables
- Ø LOBs
  - Ø BLOBs (Binary Large Object)
  - Ø CLOB (Character Large Object)

Tipos objeto

- Ø Un tipo de objeto es un tipo de dato definido por nosotros.
- Ø Promueven la reutilización.
- Ø Permiten tener un modelado más cercano a la realidad.
- Ø Tienen atributos y pueden tener métodos.

```
CREATE TYPE type_name AS OBJECT ( column_name1 column_type1
[,column_name1 column_type1,] )

CREATE TYPE POINT_TYPE AS OBJECT (X number y number)
CREATE TABLE CIRCLES (RADIUS NUMBER, CENTER POINT_TYPE)
INSERT INTO CIRCLES VALUES (6, POINT_TYPE (7, 4))

SELECT * FROM CIRCLES;
  REDIUS CENTER(X y)
  -----  ----------------
      6 POINT_TYPE (7, 4)

SELECT RADIUS, CENTER FROM CIRCLES;
  REDIUS  CENTER(X y)
  ------  ----------------
       6  POINT_TYPE(7, 4)

SELECT RADIUS, CENTER FROM CIRCLES;
  REDIUS  CENTER.X  CENTER.Y
  ------  --------  --- ----
       6         7         4

SELECT RADIUS, C.CENTER.X FROM CIRCLES C;
  REDIUS  CENTER.X
  ------  --------
       6         7
```

Tablas objeto

- Ø Es posible crear tablas donde cada renglón es un objeto.
- Ø Se crean a partir de un Tipo de Objeto.

```
CREATE TABLE table_name OF object_type

CREATE TABLE POINT OF POINT_TYPE;
     TABLE CREATED

SELECT ¡ FROM POINT:
     N) ROWS SELECTED

  INSERT INTO POINT VALUES (1, 4);
     1 ROW CREATED

  SELECT * FROM POINT;
```

```
    X     Y
---   ---
    1     4
```

Métodos

Ø Los métodos constructores son necesarios para crear un objeto y son proveídos automáticamente, pero podemos agregar otros métodos para proveer de un mayor comportamiento al objeto.
Ø En los ORDBMS son implementados como procedimientos almacenados (por ejemplo en Oracle: PL/SQL)
Ø Es necesario declararlos cuando se define el Tipo Objeto:

```
CREATE TYPE type_name AS OBJECT ( column_nameN column_typeN
MEMBER FUNCTION method_name [(argument_list)]
RETURN datatype)
```

Herencia

Ø Uno de los principales objetos de la OO es promover la reutilización.

```
CREATE TYPE type_name_subclase UNDER type_name_superclase ( column_nameN
column_typeN MEMBER FUNCTION method_name [(argument_list)]
RETURN datatype)

CREATE TYPE ARTICULO AS OBJECT ( CLAVE number, PRECIO number,
DESCRIPCION varchar2(50), ) NOT FINAL

CREATE TYPE LIBRO UNDER ARTICULO ( ISBN varchar2(10) EDITORIAL
varchar(25))

INSERT INTO LIBRO VALUES (234, 564.56, "MUS-456", "ARMONIAS", "EDITA")
```

Varrays

Ø Es un conjunto ordenado de elementos del mismo tipo.
Ø Cada elemento tiene un índice, el cual corresponde a su posición en el arreglo.
Ø Define un máximo de elementos.
Ø Permite almacenar atributos repetidos de un registro en un único renglón.
Ø Es posible crear VARRAYS a partir de Tipos de Objetos o de tipos estándar. .
Ø La información se almacenan en la tabla origen.

```
CREATE TYPE email_list AS VARRAY (10) OR VARCHAR(80);
```

Nested Tables

Ø Es un conjunto no ordenado de elementos.
Ø No tiene un máximo de elementos determinado.
Ø Es posible seleccionar, insertar, borrar y actualizar como en una tabla ordinaria.
Ø Se almacenan por separado de la tabla origen.

Large Objects

Ø Un gran objeto (LOB - Large Object) almacena grandes volúmenes de datos. .
Ø BLOB
  Ø Para datos binarios (soporta hasta 4GB).
  Ø Se almacena dentro de la base de datos.
Ø CLOB
  Ø Almacena caracteres (más de 4 GB)
  Ø Se almacena dentro de la base de datos.
Ø BFILE
  Ø Es un puntero a un archivo externo.
  Ø Los archivos referenciados existen en el sistema de archivos; la base de datos sólo mantiene el puntero.
  Ø El tamaño es limitado por el sistema operativo.

OR - DBMS's

Ø Oracle 8, 9, 10g .
Ø Informix .
Ø DB2 .
Ø Postgress

Capítulo VII. Usos Avanzados de las Bases de Datos

Datamining

La tecnología informática constituye la infraestructura fundamental de las grandes organizaciones y permite registrar múltiples detalles de la vida de las empresas. Las bases de datos posibilitan almacenar cada transacción, así como otros muchos elementos que reflejan la interacción de la organización con otras organizaciones, clientes o internamente, entre sus divisiones y empleados, etcétera.

Es imprescindible convertir los grandes volúmenes de datos existentes en experiencia, conocimiento y sabiduría, formas que atesora la humanidad para que sea útil a la toma de decisiones, especialmente en las grandes organizaciones y proyectos científicos. La búsqueda de información relevante siempre es útil a la administración empresarial; el control de la producción, el análisis de los mercados, el diseño en ingeniería y la exploración científica, porque pueden ofrecer las respuestas más apropiadas a las necesidades de información. Varias preguntas se relacionan frecuentemente con los datos, la información y el conocimiento. Su respuesta, demanda la participación de varios especialistas. ¿Cómo puede entenderse un fenómeno sobre la base de la interpretación de grandes volúmenes de datos? ¿De qué manera puede utilizarse la información para la toma de decisiones?, son algunos ejemplos de interrogantes comunes.

La respuesta a estas preguntas es el objetivo del DM, un conjunto de técnicas agrupadas con el fin de crear mecanismos adecuados de dirección, entre ellas puede citarse la estadística, el reconocimiento de patrones, la clasificación y la predicción.

Para descubrir patrones de relaciones útiles en un conjunto de datos se empezaron a utilizar métodos que fueron denominados de diferente forma. El término data mining en inglés no era al principio del agrado de muchos estadísticos, porque sus investigaciones estaban dirigidas a procesar y reprocesar suficientemente los datos, hasta que confirmasen o refutasen las hipótesis planteadas. Desde este ángulo, la minería de datos aplica una dinámica que se mueve en sentido contrario al método científico tradicional. Con frecuencia, el investigador formula una hipótesis; luego, diseña un experimento para captar los datos necesarios y realizar los experimentos que confirmen o refuten la hipótesis planteada. Este es un proceso, que realizado de forma rigurosa, debe generar nuevos conocimientos. En la minería de datos, por el contrario, se captan y procesan los datos con la esperanza de que de ellos surja una hipótesis apropiada. Se desea que los datos nos describan o indiquen el porqué presentan determinada configuración y comportamiento.

El DM es un mecanismo de explotación, consistente en la búsqueda de información valiosa en grandes volúmenes de datos. Está muy ligada a los Data Warehousing que proporcionan la información histórica con la cual los algoritmos de minería de datos tienen la información necesaria para la toma de decisiones.

La minería de datos (DM) y el descubrimiento de conocimientos en bases de datos (KDD)

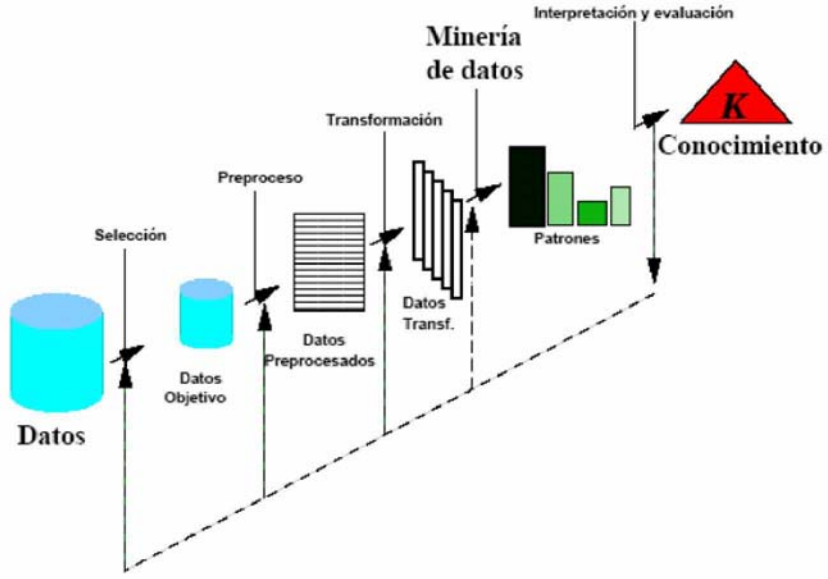
Existe cierta tendencia a identificar como sinónimos a la minería de datos y el descubrimiento de conocimientos en bases de datos, que de forma abreviada se refiere con las siglas KDD (del inglés Knowledge Discovery in Data Bases), la convergencia del aprendizaje automático, la estadística, el reconocimiento de patrones, la inteligencia artificial, las bases de datos, la visualización de datos, los sistemas para el apoyo a la toma de decisiones, la recuperación de información y otros muchos campos.

El KDD es el proceso completo de extracción de conocimientos, no trivial, previamente desconocidos y potencialmente útil a partir de un conjunto de datos, mientras que “la

minería de datos es una compilación de técnicas reunidas para crear mecanismos adecuados para la toma de decisiones. Entre estas técnicas se pueden citar la estadística, el reconocimiento de patrones, la clasificación y la predicción, la excavación de información relevante de la administración empresarial, el control de la producción, el análisis de los mercados, el diseño en ingeniería y la exploración científica." En otras palabras, el concepto minería de datos se asocia al proceso de construcción de reglas a partir de colecciones de datos con una finalidad previamente determinada y para su uso en la toma de decisiones con respecto a dicha finalidad. El concepto de KDD no comprende necesariamente esta segunda parte. Esta diferencia, muchas veces inadvertida, puede ser la causa de que ambos conceptos se utilicen indistintamente en gran parte de la literatura.

KDD = proceso completo: Extracción no trivial de conocimiento implícito, previamente desconocido y potencialmente útil, a partir de una base de datos. (Frawley et al., 1991).

DM = etapa de descubrimiento en el proceso de KDD: Paso consistente en el uso de algoritmos concretos que generan una enumeración de patrones a partir de los datos preprocesados. (Fayyad et 1996).



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 DM.
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 DM.
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.
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.

Fundamentos del Data Mining.

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. DM toma este proceso de evolución más allá del acceso y navegación retrospectiva de los datos, hacia la entrega de información prospectiva y proactiva. 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

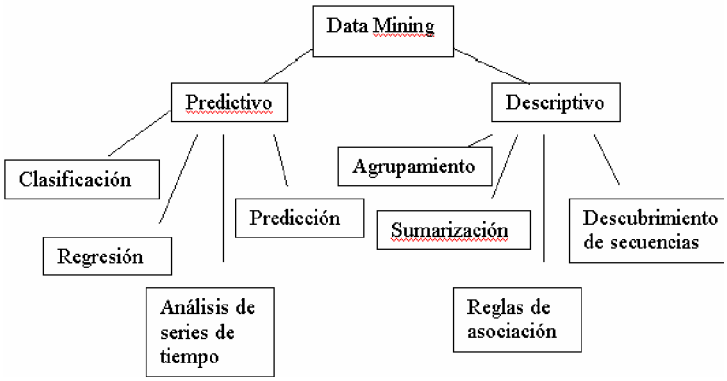
Los componentes esenciales de la tecnología de DM han estado bajo desarrollo por décadas, en áreas de investigación como estadísticas, inteligencia artificial y aprendizaje de máquinas. Hoy, la madurez de estas técnicas, junto con los motores de bases de datos relacionales de alta performance, hicieron que estas tecnologías fueran prácticas para los entornos de Data Warehouse actuales.

El DM envuelve muchos algoritmos para resolver diferentes tareas. Todos estos algoritmos intentan estar en un modelo de datos. El algoritmo examina los datos y determina un modelo que es cerrado. Los algoritmos de DM pueden ser caracterizados consistiendo en tres partes:

Ø Modelo. El propósito del algoritmo es el de colocar el dato en un modelo.
Ø Preferencia. Algún criterio debe ser usado para colocar un modelo en otro.
Ø 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. Las tareas de este modelo incluyen la clasificación, regresión, análisis de series de tiempo y predicción.

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. El agrupamiento (clustering), sumarización, reglas de asociación y descubrimiento de secuencias son vistas como tareas del modelo descriptivo.

Tareas Básicas del Data Mining

Clasificación. Traza mapas de datos dentro de grupos o clases predefinidos. Esto se obtiene refiriéndose a como supervisar el aprendizaje, porque las clases son determinadas antes de examinar los datos. Los algoritmos de clasificación requieren que las clases sean definidas basadas sobre los valores de los atributos de los datos. Ellos obtienen la descripción de estas clases mirando las características de los datos conocidos existentes que pertenecen a las clases.

Regresión. Ésta es usa para un mapa de un detalle de un dato para la predicción de un valor real de una variable. En la actualidad, la regresión envuelve el aprendizaje de la función que hace este mapeo. La regresión asume que la meta del dato se encuentre dentro de un tipo de función conocida (ejemplo: lineal, logística, etc.) y entonces determina la mejor función de este tipo que los modelos dan al dato.

Análisis de series de tiempo. Con el análisis de series de tiempo, el valor de un atributo es examinado de como varia éste sobre el tiempo. Los valores usualmente son obtenidos como sucesos separados por puntos en el tiempo (diariamente, semanalmente, etc.).

Predicción. Muchas aplicaciones de DM del mundo real se pueden ver, prediciendo el dato futuro basado sobre los estados pasado y actual del dato. La predicción puede ser vista como un tipo de clasificación. Cabe notar que ésta es una tarea de DM que es diferente del modelo predictivo, aunque la tarea de predicción es un tipo del modelo predictivo. Las

aplicaciones de predicción incluyen el conocimiento del habla, aprendizaje de la máquina, reconocimiento de patrones, inundamientos.

Agrupamiento (Clustering). Es un Proceso de dividir un conjunto de datos en grupos mutuamente excluyentes de tal manera que cada miembro de un grupo esté lo "más cercano" posible a otro y grupos diferentes estén lo "más lejos" posible uno del otro, donde la distancia está medida con respecto a todas las variables disponibles.

Sumarización. Son mapas de datos dentro de subseries o subconjuntos con simples descripciones asociadas. Ésta es llamada también caracterización o generalización. Ésta extrae o deriva información descriptiva acerca de la base de datos.

Reglas de asociación. Establece asociaciones en base a los perfiles de los clientes sobre los cuales se está realizando el DM. Las reglas de Asociación están siempre definidas sobre atributos binarios. No es muy complicado generar reglas en grandes bases de datos. El problema es que tal algoritmo eventualmente puede dar información que no es relevante. DM envuelve modelos para determinar patrones a partir de los datos observados. Los modelos juegan un rol de conocimiento inferido. Diciendo cuando el conocimiento representa conocimiento útil o no, esto es parte del proceso de extracción de conocimiento en bases de datos (Knowledge Discovery in Databases - KDD).

Descubrimiento de secuencias. El descubrimiento de secuencias o análisis secuencial es usado para determinar patrones secuenciales en los datos. Estos patrones son basados sobre una secuencia de acciones en el tiempo. Los patrones son similares a las asociaciones, los datos o eventos son encontrados para ser relatados, pero la relación es basada en el tiempo.

Técnicas del Data Mining

Redes neuronales artificiales. Modelos predecibles no - lineales que aprenden a través del entrenamiento y semejan la estructura de una red neuronal biológica.

Árboles de decisión. Estructuras de forma de árbol que representan conjuntos de decisiones. Estas decisiones generan reglas para la clasificación de un conjunto de datos. Métodos específicos de árboles de decisión incluyen Árboles de Clasificación y Regresión (CART: Classification And Regression Tree) y Detección de Interacción Automática de Chi Cuadrado (CHAI: Chi Square Automatic Interaction Detection):

- Ø CART. Técnica usada para la clasificación de un conjunto da datos. Provee un conjunto de reglas que se pueden aplicar a un nuevo (sin clasificar) conjunto de datos para predecir cuáles registros darán un cierto resultado. Segmenta un conjunto de datos creando 2 divisiones. Requiere menos preparación de datos que CHAID.
- Ø CHAID. Técnica similar a la anterior, pero segmenta un conjunto de datos utilizando tests de chi cuadrado para crear múltiples divisiones.

Algoritmos genéticos. Técnicas de optimización que usan procesos tales como combinaciones genéticas, mutaciones y selección natural en un diseño basado en los conceptos de evolución.

Método del vecino más cercano. Una técnica que clasifica cada registro en un conjunto de datos basado en una combinación de las clases de los k registros más similares a él en un conjunto de datos históricos. Algunas veces se llama la técnica del vecino≥ k - más cercano.

Regla de inducción. La extracción de reglas if - then de datos basados en significado estadístico.

Extensiones del Data Mining

Web mining. Consiste en aplicar las técnicas de minería de datos a documentos y servicios del Web. Todos los que visitan un sitio en Internet dejan huellas digitales (direcciones de IP, navegador, etc.) que los servidores automáticamente almacenan en una bitácora de accesos (Log). Las herramientas de Web mining analizan y procesan estos logs para producir información significativa. Debido a que los contenidos de Internet consisten en varios tipos de datos, como texto, imagen, vídeo, metadatos o hiperligas, investigaciones recientes usan el término multimedia data mining (minería de datos multimedia) como una instancia del Web mining para tratar ese tipo de datos. Los accesos totales por dominio, horarios de accesos

más frecuentes y visitas por día, entre otros datos, son registrados por herramientas estadísticas que complementan todo el proceso de análisis del Web mining.

Text mining. Dado que el ochenta por ciento de la información de una compañía está almacenada en forma de documentos, las técnicas como la categorización de texto, el procesamiento de lenguaje natural, la extracción y recuperación de la información o el aprendizaje automático, apoyan al text mining. En ocasiones se confunde el text mining con la recuperación de la información (Information Retrieval o IR). Esta última consiste en la recuperación automática de documentos relevantes mediante indexaciones de textos, clasificación, categorización, etc. Generalmente se utilizan palabras clave para encontrar una página relevante. En cambio, el text mining se refiere a examinar una colección de documentos y descubrir información no contenida en ningún documento individual de la colección; en otras palabras, trata de obtener información sin haber partido de algo.

Datawarehousing

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, debido a lo cual se ha convertido actualmente en una de las tendencias tecnológicas más significativas en la administración de información.

Los almacenes de datos (Datawarehouse) generan bases de datos tangibles con una perspectiva histórica, utilizando datos de múltiples fuentes que se fusionan en forma congruente. Estos datos se mantienen actualizados, pero no cambian al ritmo de los sistemas transaccionales. Muchos DW se diseñan para contener un nivel de detalle hasta el nivel de transacción, con la intención de hacer disponible todo tipo de datos y características, para reportar y analizar. Así un DW resulta ser un recipiente de datos transaccionales para proporcionar consultas operativas y la información para poder llevar a cabo análisis multidimensional. De esta forma, dentro de un almacén de datos existen dos tecnologías complementarias, una relacional para consultas y una multidimensional para análisis.

Existen muchas definiciones para el DW, la más conocida fue propuesta por Inmon[MicroSt96] (considerado el padre de las Bases de Datos) en 1992: “Un DW es una colección de datos orientados a temas, integrados, no-volátiles y variante en el tiempo, organizados para soportar necesidades empresariales”. En 1993, Susan Osterfeldt[MicroSt96] 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.

DW se sustenta en un procesamiento distinto al utilizado por los sistemas operacionales, OLAP (Procesamiento Analítico en Línea), el cual surge como un proceso para ser usado en el análisis de negocios y otras aplicaciones que requieren una visión flexible del negocio.

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 Finanzas o Marketing. 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. Sin embargo, muchos desarrolladores se dieron cuenta de que estas aplicaciones eran

susceptibles de ser generalizadas y servir para casi cualquier necesidad, esto es, para casi cualquier base de datos. Los sistemas OLAP evitan la necesidad de desarrollar interfaces de consulta y ofrecen un entorno único valido 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.

Para los desarrolladores de aplicaciones acostumbrados a trabajar con bases de datos relacionales, el diseño de una base de datos multidimensional puede ser complejo. Pero en general, nuestra experiencia nos dice que el diseño de dimensiones y variables es mucho más sencillo e intuitivo que un diseño relacional. Esto es debido a que las dimensiones y variables son reflejo directo de los informes en papel utilizados por la organización.

Una vez que se ha decidido emplear un entorno de consulta OLAP, se ha de elegir entre R-OLAP y 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. Es decir, un sistema M-OLAP contiene precalculados (almacenados) los resultados de todas las posibles consultas a la base de datos.

M-OLAP consigue consultas muy rápidas a costa de mayores necesidades de almacenamiento y retardos en las modificaciones (que no deberían producirse salvo excepcionalmente) y largos procesos batch de carga y cálculo de acumulados. En R-OLAP, al contener sólo las combinaciones de valores de dimensión que representan detalle, es decir, al no haber redundancia, el archivo de base de datos es pequeño. Los procesos batch de carga son rápidos (ya que no se requiere agregación) y sin embargo, las consultas pueden ser muy lentas, por lo que se aplica la solución de tener al menos algunas consultas precalculadas.

En M-OLAP, el gran tamaño de las variables multidimensionales o el retardo en los procesos batch puede ser un inconveniente.

Características de un Datawarehouse

Orientado a temas. Una primera característica del DW es que la información se clasifica en base a los aspectos que son de interés para la empresa. Siendo así, los datos tomados están en contraste con los clásicos procesos orientados a las aplicaciones.

Integración. El aspecto más importante del ambiente DW es que la información encontrada al interior está siempre integrada. La integración de datos se muestra de muchas maneras: en convenciones de nombres consistentes, en la medida uniforme de variables, en la codificación de estructuras consistentes, en atributos físicos de los datos consistentes, fuentes múltiples y otros.

De tiempo variante.
Toda la información del DW es requerida en algún momento. Esta característica básica de los datos en un depósito, es muy diferente de la información encontrada en el ambiente operacional. En éstos, la información se requiere al momento de acceder. En otras palabras, en el ambiente operacional, cuando usted accede a una unidad de información, usted espera que los valores requeridos se obtengan a partir del momento de acceso.

Como la información en el DW es solicitada en cualquier momento (es decir, no "ahora mismo"), los datos encontrados en el depósito se llaman de "tiempo variante".

Los datos históricos son de poco uso en el procesamiento operacional. La información del depósito por el contraste, debe incluir los datos históricos para usarse en la identificación y evaluación de tendencias.

El tiempo variante se muestra de varias maneras:

- La más simple es que la información representa los datos sobre un horizonte largo de tiempo - desde cinco a diez años. El horizonte de tiempo representado para el ambiente operacional es mucho más corto - desde valores actuales hasta sesenta a noventa días. Las aplicaciones que tienen un buen rendimiento y están disponibles para el

procesamiento de transacciones, deben llevar una cantidad mínima de datos si tienen cualquier grado de flexibilidad. Por ello, las aplicaciones operacionales tienen un corto horizonte de tiempo, debido al diseño de aplicaciones rígidas.

- Ø La segunda manera en la que se muestra el tiempo variante en el DW está en la estructura clave. Cada estructura clave en el DW contiene, implícita o explícitamente, un elemento de tiempo como día, semana, mes, etc. El elemento de tiempo está casi siempre al pie de la clave concatenada, encontrada en el DW. En ocasiones, el elemento de tiempo existirá implícitamente, como el caso en que un archivo completo se duplica al final del mes o al cuarto.
- Ø La tercera manera en que aparece el tiempo variante es cuando la información del DW, una vez registrada correctamente, no puede ser actualizada. La información del DW es, para todos los propósitos prácticos, una serie larga de "snapshots" (vistas instantáneas).

No volátil. La información es útil sólo cuando es estable. Los datos operacionales cambian sobre una base momento a momento. La perspectiva más grande, esencial para el análisis y la toma de decisiones, requiere una base de datos estable.

Estructura del Datawarehouse

Los DW tienen una estructura distinta. Hay niveles diferentes de esquematización y detalle que delimitan el DW.

Detalle de datos actuales.
En gran parte, el interés más importante radica en el detalle de los datos actuales, debido a que:

- Ø Refleja las ocurrencias más recientes, las cuales son de gran interés
- Ø Es voluminoso ya que se almacena al más bajo nivel de granularidad.

Casi siempre se almacena en disco, el cual es de fácil acceso, aunque su administración sea costosa y compleja.

Detalle de datos antiguos. La data antigua es aquella que se almacena sobre alguna forma de almacenamiento masivo. No es frecuentemente su acceso y se almacena a un nivel de detalle, consistente con los datos detallados actuales. Mientras no sea prioritario el almacenamiento en un medio de almacenaje alterno, a causa del gran volumen de datos unido al acceso no frecuente de los mismos, es poco usual utilizar el disco como medio de almacenamiento.

Datos ligeramente resumidos.
La data ligeramente resumida es aquella que proviene desde un bajo nivel de detalle encontrado al nivel de detalle actual. Este nivel del DW casi siempre se almacena en disco. Los puntos en los que se basa el diseñador para construirlo son:

- Ø Que la unidad de tiempo se encuentre sobre la esquematización hecha.
- Ø Qué contenidos (atributos) tendrá la data ligeramente resumida.

A veces se encuentra en el ambiente de DW y en otros, fuera del límite de la tecnología que ampara al DW. (De todos modos, los datos completamente resumidos son parte del DW sin considerar donde se alojan los datos físicamente).

Metadata.
El componente final del DW es el de la metadata. De muchas maneras la metadata se sitúa en una dimensión diferente al de otros datos del DW, debido a que su contenido no es tomado directamente desde el ambiente operacional. La metadata juega un rol especial y muy importante en el DW y es usada como:

- Ø Un directorio para ayudar al analista a ubicar los contenidos del DW.
- Ø Una guía para la trazabilidad de los datos, de cómo se transforma, del ambiente operacional al de DW.
- Ø Una guía de los algoritmos usados para la esquematización entre el detalle de datos actual, con los datos ligeramente resumidos y éstos, con los datos completamente resumidos, etc.

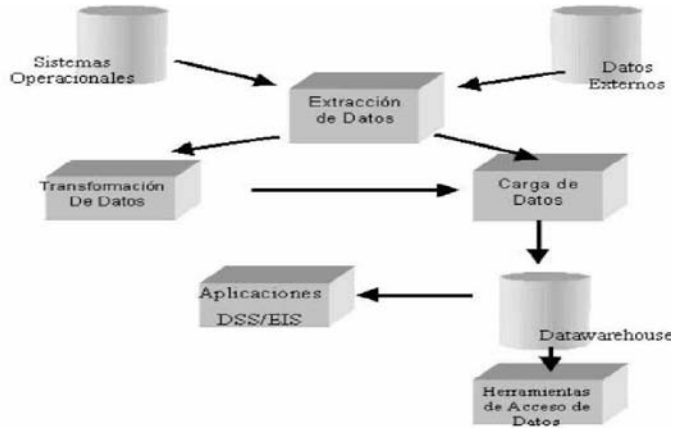
Arquitectura de un Datawarehouse

Componentes y Estructuras.
El término DataWarehouse se utiliza indistintamente para hablar de la arquitectura en sí como también para uno de los componentes que la conforman, específicamente el que tiene relación con el almacenamiento físico de los datos.

La estructura básica de la arquitectura DW incluye:

- Ø Datos operacionales: un origen de datos para el componente de almacenamiento físico DW.
- Ø Extracción de Datos: selección sistemática de datos operacionales usados para poblar el componente de almacenamiento físico DW.
- Ø Transformación de datos. Procesos para sumarizar y realizar otros cambios en los datos operacionales para reunir los objetivos de orientación a temas e integración principalmente.
- Ø Carga de Datos: inserción sistemática de datos en el componente de almacenamiento físico DW.
- Ø Datawarehouse: almacenamiento físico de datos de la arquitectura DW.
- Ø Herramientas de Acceso al componente de almacenamiento físico DW:

Herramientas que proveen acceso a los datos.



Transformación de Datos y Metadata

Transformación de datos. Uno de los desafíos de cualquier implementación de DW, es el problema de transformar los datos. La transformación se encarga de las inconsistencias en los formatos de datos y la codificación, que pueden existir dentro de una base de datos única y que casi siempre existen cuando múltiples bases de datos contribuyen al DW.

Metadata

Otro aspecto de la arquitectura de DW es crear soporte a la metadata. Metadata es la información sobre los datos que se alimenta, se transforma y existe en el DW. Metadata es un concepto genérico, pero cada implementación de la metadata usa técnicas y métodos específicos.

Estos métodos y técnicas son dependientes de los requerimientos de cada organización, de las capacidades existentes y de los requerimientos de interfaces de usuario. Hasta ahora, no hay normas para la metadata, por lo que la metadata debe definirse desde el punto de vista del software DW, seleccionado para una implementación específica.

Típicamente, la metadata incluye los siguientes puntos:

1. Las estructuras de datos que dan una visión de los datos al administrador de datos.
2. Las definiciones del sistema de registro desde el cual se construye el DW.
3. Las especificaciones de transformaciones de datos que ocurren tal como la fuente de datos se replica al DW.

El modelo de datos del DW (es decir, los elementos de datos y sus relaciones). Un registro de cuando los nuevos elementos de datos se agregan al DW y cuando los elementos de datos antiguos se eliminan o se resumen. Los niveles de sumarización, el método de sumarización y las tablas de registros de su DW.

Algunas implementaciones de la metadata también incluyen definiciones de la(s) vista(s) presentada(s) a los usuarios del DW. Típicamente, se definen vistas múltiples para favorecer las preferencias variadas de diversos grupos de usuarios. En otras implementaciones, estas descripciones se almacenan en un Catálogo de Información.

Los esquemas y subesquemas para bases de datos operacionales, forman una fuente óptima de entrada cuando se crea la metadata. Hacer uso de la documentación existente,

especialmente cuando está disponible en forma electrónica, puede acelerar el proceso de definición de la metadata del ambiente DW.

La metadata sirve, en un sentido, como el corazón del ambiente DW. Crear definiciones de metadata completa y efectiva puede ser un proceso que consuma tiempo, pero lo mejor de las definiciones y si usted usa herramientas de gestión de software integrado, son los esfuerzos que darán como resultado el mantenimiento del DW.

Flujo de Datos

Existe un flujo de datos normal y predecible dentro del DW. Los datos ingresan al data warehouse desde el ambiente operacional. (Hay pocas excepciones a esta regla).

Al ingresar al data warehouse, la información va al nivel de detalle actual, tal como se muestra. Se queda allí y se usa hasta que ocurra uno de los tres eventos siguientes:
Ø Sea eliminado
Ø Sea resumido
Ø Sea archivado

Con el proceso de desactualización en un DW se mueve el detalle de la data actual a data antigua, basado en el tiempo de los datos. El proceso de esquematización usa el detalle de los datos para calcular los datos en forma ligera y completamente resumidos. Hay pocas excepciones al flujo mostrado. Sin embargo, en general, para la mayoría de datos encontrados en un DW, el flujo de la información es como se ha explicado.

Medios de Almacenamiento para Información Antigua

Hay una amplia variedad de medios de almacenamiento que deben considerarse para almacenar datos más antiguos. Dependiendo del volumen de información, la frecuencia de acceso, el costo de los medios y el tipo de acceso, es probable que otros medios de almacenamiento sirvan a las necesidades del nivel de detalle más antiguo en el DW.
Ø Almacenamiento fotoóptico
Ø Raid
Ø Microficha
Ø Cinta magnética
Ø Almacenamiento en masa

Base de Datos Multidimensionales

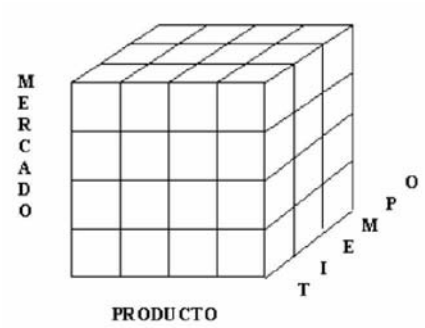
El Modelamiento Dimensional es una técnica para modelar bases de datos simples y entendibles al usuario final. La idea fundamental es que el usuario visualice fácilmente la relación que existe entre las distintas componentes del modelo. Consideremos un punto en el espacio. El espacio se define a través de sus ejes coordenados (por ejemplo X, Y y Z). Un punto cualquiera de este espacio quedará determinado por la intersección de tres valores particulares de sus ejes.

Si se le asignan valores particulares a estos ejes. Digamos que el eje X representa Productos, el eje Y representa el Mercado y el eje Z corresponde al Tiempo. Se podría tener, la siguiente combinación: producto = madera, mercado = Concepción, tiempo = diciembre-1998. La intersección de estos valores nos definirá un solo punto en nuestro espacio. Si el punto que buscamos, lo definimos como la cantidad de madera vendida, entonces se tendrá un valor específico y único para tal combinación.

En el modelo multidimensional cada eje corresponde a una dimensión particular. Entonces la dimensionalidad de nuestra base estará dada por la cantidad de ejes (o dimensiones) que le asociemos.

Cuando una base puede ser visualizada como un cubo de tres o más dimensiones es más fácil para el usuario organizar la información e imaginarse en ella cortando y rebanando el cubo a través de cada una de sus dimensiones, para buscar la información deseada.

Esto puede visualizarse como un cubo, donde cada punto dentro del cubo es una intersección de coordenadas definidas por los lados de éste (dimensiones). Ejemplos de medidas son: unidades producidas, unidades vendidas, costo de unidades producidas, ganancias ($) de unidades vendidas, etc.



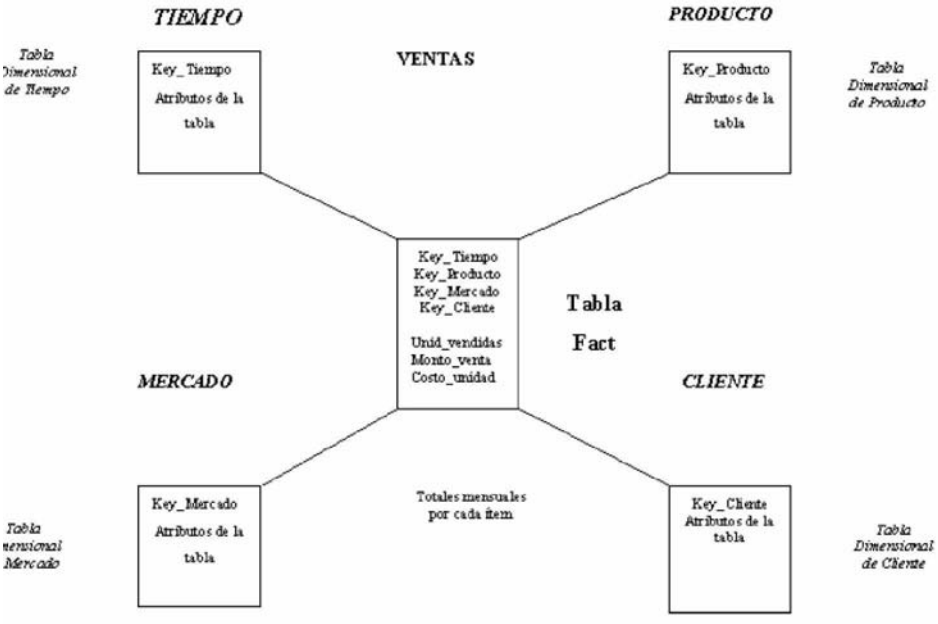
Características del Modelo Multidimensional

En general, la estructura básica de un DW para el Modelo Multidimensional está definida por dos elementos: esquemas y tablas.

- Ø Tablas DW: como cualquier base de datos relacional, un DW se compone de tablas. Hay dos tipos básicos de tablas en el Modelo Multidimensional:
  - Ø Tablas Fact: contienen los valores de las medidas de negocios, por ejemplo: ventas promedio en dólares, unidades vendidas, etc.
  - Ø Tablas Lock_up: contienen el detalle de los valores que se encuentran asociados a la tabla Fact.
- Ø Esquemas DW: la colección de tablas en el DW se conoce como Esquema. Los esquemas caen dentro de dos categorías básicas: esquemas estrellas y esquemas snowflake.

Esquema de Estrella.

En general, el modelo multidimensional también se conoce con el nombre de esquema estrella, pues su estructura base es similar: una tabla central y un conjunto de tablas que la atienden radialmente. El esquema estrella deriva su nombre del hecho que su diagrama forma una estrella, con puntos radiales desde el centro. El centro de la estrella consiste de una o más tablas fact y las puntas de la estrella son las tablas lock_up. Este modelo entonces, resulta ser asimétrico, pues hay una tabla dominante en el centro con varias conexiones a las otras tablas. Las tablas Lock-up tienen sólo la conexión a la tabla fact y ninguna más.



Esquema Snowflake.

La diferencia del esquema snowflake comparado con el esquema estrella, está en la estructura de las tablas lock_up: las tablas lock_up en el esquema snowflake están normalizadas. Cada tabla lock_up contiene sólo el nivel que es clave primaria en la tabla y la foreign key de su parentesco del nivel más cercano del diagrama.

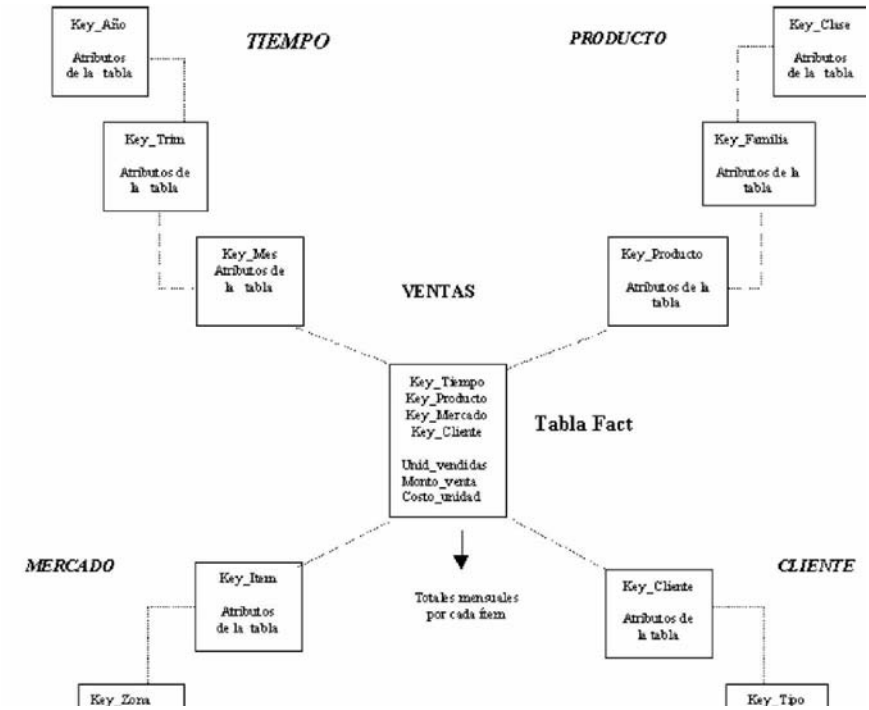


Tabla Fact o de Hechos.
Es la tabla central en un esquema dimensional. Es en ella donde se almacenan las mediciones numéricas del negocio. Estas medidas se hacen sobre la unidad básica de la tabla.

El grano o la granularidad de la tabla queda determinada por el nivel de detalle que se almacenará en la tabla. Por ejemplo, para el caso de producto, mercado y tiempo, el grano puede ser la cantidad de madera vendida 'mensualmente'. El grano revierte las unidades atómicas en el esquema dimensional.

Cada medida es tomada de la intersección de las dimensiones que la definen. Idealmente está compuesta por valores numéricos, continuamente evaluados y aditivos. La razón de estas características es que así se facilita que los miles de registros que involucran una consulta sean comprimidos en unas pocas líneas en un set de respuesta.

La clave de la tabla fact recibe el nombre de clave compuesta o concatenada debido a que se forma de la composición (o concatenación) de las llaves primarias de las tablas dimensionales a las que está unida.

Así entonces, se distinguen dos tipos de columnas en una tabla fact: columnas fact y columnas key. Donde la columna fact es la que almacena alguna medida de negocio y una columna key forma parte de la clave compuesta de la tabla.

Tablas Lock-up o Dimensionales.
Estas tablas son las que se conectan a la tabla fact, son las que alimentan a la tabla fact. Una tabla lock_up almacena un conjunto de valores que están relacionados a una dimensión particular. Tablas lock_up no contienen hechos, en su lugar los valores en las tablas lock_up son los elementos que determinan la estructura de las dimensiones. Así entonces, en ellas existe el detalle de los valores de la dimensión respectiva.

Una tabla lock_up está compuesta de una primary key que identifica unívocamente una fila en la tabla junto con un conjunto de atributos y dependiendo del diseño del modelo multidimensional puede existir una foreign key que determina su relación con otra tabla lock_up.

Para decidir si un campo de datos es un atributo o un hecho se analiza la variación de la medida a través del tiempo. Si varía continuamente implicaría tomarlo como un hecho, caso contrario será un atributo.

Los atributos dimensionales son un rol determinante en un DDW. Ellos son la fuente de todas las necesidades que debieran cubrirse. Esto significa que la base de datos será tan buena como lo sean los atributos dimensionales, mientras más descriptivos, manejables y de buena calidad, mejor será el DDW.

Pasos Básicos del modelamiento Multidimensional

Decidir cuáles serán los procesos de negocios a modelar, basándose en el conocimiento de éstos y de los datos disponibles. Ejemplo: Gastos realizados por cada mercado para cada ítem a nivel mensual. Productos vendidos por cada mercado según el precio en cada mes.

Decidir el Grano de la tabla Fact de cada proceso de negocio. Ejemplo: Producto x mercado x tiempo. En este punto se debe tener especial cuidado con la magnitud de la base de datos, con la información que se tiene y con las preguntas que se quiere responder. El grano decidirá las dimensiones del DDW. Cada dimensión debe tener el grano más pequeño que se pueda puesto que las preguntas que se realicen necesitan cortar la base en caminos precisos (aunque las preguntas no lo pidan explícitamente).

Decidir las dimensiones a través del grano. Las dimensiones presentes en la mayoría de los DDW son: tiempo, mercado, producto, cliente. Un grano bien elegido determina la dimensionalidad primaria de la tabla fact. Es posible usualmente agregar dimensiones adicionales al grano básico de la tabla fact, donde estas dimensiones adicionales toman un solo valor para cada combinación de las dimensiones primarias. Si se reconoce que una dimensión adicional deseada viola el grano por causar registros adicionales a los generados, entonces el grano debe ser revisado para acomodar esta dimensión adicional.

Elegir las mediciones del negocio para la tabla fact. Se deben establecer los ítemes que quedarán determinados por la clave compuesta de la tabla fact.

Profundizaciones de Diseño

La Dimensión Tiempo.
Virtualmente se garantiza que cada DDW tendrá una tabla dimensional de tiempo, debido a la perspectiva de almacenamiento histórica de la información. Usualmente es la primera dimensión en definirse, con el objeto de establecer un orden ya que la inserción de datos en la base de datos multidimensional se hace por intervalos de tiempo, lo cual asegura un orden implícito.

Dimensiones que varían lentamente en el tiempo.
Son aquellas dimensiones que se mantienen “casi” constantes en el tiempo y que pueden preservar la estructura dimensional independiente del tiempo, con sólo agregados menores relativos para capturar la naturaleza cambiante del tiempo.
Cuando se encuentra una de estas dimensiones se está haciendo una de las siguientes tres elecciones fundamentales. Cada elección resulta en un diferente grado de seguimiento sobre el tiempo:

- Tipo 1: Sobrescribir el viejo valor en el registro dimensional y por lo tanto perder la capacidad de seguir la vieja historia.
- Tipo 2: Crear un registro dimensional adicional (con una nueva llave) que permita registrar el cambio presentado por el valor del atributo. De esta forma permanecerían en la base tanto el antiguo como el nuevo valor del registro con lo cual es posible segmentar la historia de la ocurrencia.
- Tipo 3: Crear un campo “actual” nuevo en el registro dimensional original el cual almacene el valor del nuevo atributo, manteniendo el atributo original también. Cada vez que haya un nuevo cambio en el atributo, se modifica el campo “actual” solamente. No se mantiene un registro histórico de los cambios intermedios.

Niveles.
Un nivel representa un nivel particular de agregación dentro de una dimensión; cada nivel sobre el nivel base representa la sumarización total de los datos desde el nivel inferior. Por ejemplo: consideremos una dimensión Tiempo con tres niveles: Mes, Semestre, Año. El nivel Mes representa el nivel base, el nivel Semestre representa la sumarización de los totales por Mes y el nivel Año representa la sumarización de los totales para los Semestres.

Agregar niveles de sumarización otorga flexibilidad adicional a usuarios finales de aplicaciones EIS/ DSS para analizar los datos.

Sobre Jerarquías.
A nivel de dimensiones es posible definir jerarquías, las cuales son grupos de atributos que siguen un orden preestablecido.

Una jerarquía implica una organización de niveles dentro de una dimensión, con cada nivel representando el total agregado de los datos del nivel inferior. Las jerarquías definen cómo los datos son sumarizados desde los niveles más bajos hacia los más altos. Una dimensión

típica soporta una o más jerarquías naturales. Una jerarquía puede pero no exige contener todos los valores existentes en la dimensión.

Se debe evitar caer en la tentación de convertir en tablas dimensionales separadas cada una de las relaciones muchos-a-uno presentes en las jerarquías. Esta descomposición es irrelevante en el planeamiento del espacio ocupado en disco y sólo dificulta el entendimiento de la estructura para el usuario final, además de destruir el desempeño del browsing.

Base de Datos Inteligentes

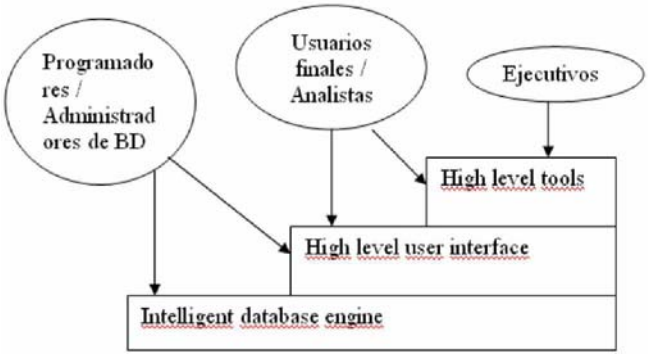
Base De Datos Inteligentes

Las bases de datos inteligentes representan una tecnología para la administración de la información que fue envuelta como resultado de la integración de las bases de datos tradicionales con otros campos como lo son:

- Ø Programación orientada a objetos.
- Ø Sistemas expertos.
- Ø Hypermedia.
- Ø Administrador de Textos.
- Ø Tecnología de Bases de Datos Tradicional.
- Ø Recepción de información en línea.

La arquitectura de las bases de datos inteligentes consta de tres niveles, los cuales son:

- Ø High-level tools
- Ø High-level user interface
- Ø Intelligent



Estas herramientas proveen al usuario muchas facilidades como son búsquedas inteligentes, calidad de los datos y control de integridad, además de descubrimiento autómata. Estas herramientas representan una biblioteca externa de herramientas poderosas que algunos usuarios quizás encuentren útiles y otros no. Muchas de éstas pueden ser clasificadas como técnicas de administración para la información.

High-level tools

Se pueden obtener siete tipos de herramientas High-level tools, las cuales son:

- Ø Herramientas de descubrimiento de conocimiento. Esta categoría incluye herramientas para el análisis de datos, aprendizaje de la máquina y análisis estadísticos. Estas herramientas representan una nueva y excitante frontera en la tecnología de base de datos inteligentes que nos permite extraer conocimiento desde datos.
- Ø Herramientas de integridad de datos y control de calidad. Son necesarias debido a los no deseados efectos del crecimiento en número y tamaño de las bases de datos.
- Ø Herramientas de administración de hipermedia. Estas permiten a los desarrolladores y usuarios construir sistemas de información hipermedia que combinan una forma libre de colocar texto, dato, imágenes, sonido, etc. Esta categoría refleja el hecho que la información quizás será expresada en muchas diferentes formas o medios y que métodos son necesarios para organizar y acceder a estas diferentes formas de información.
- Ø Herramientas de presentación y despliegue de los datos. Dando una gran base de datos, muchos usuarios desean ver y desplegar alguno de los datos o resúmenes. Proveen gráficos, formas y otros tipos de presentación de datos.
- Ø Herramientas de análisis de escenarios y de soporte de decisiones. Permiten una fusión uniforme entre las hojas de cálculo, bases de datos, sistemas de modelado financiero, etc.
- Ø Herramientas de administración de formato de datos. Permiten a los usuarios la transformación entre los formatos de datos, por ejemplo, la fusión de un archivo ASCII

con un dato en formato dBASE y datos obtenidos desde una base de datos DB2 generados automáticamente con un query SQL. Estas herramientas son indispensables para aplicación que confíe sobre el análisis real de datos.

- Herramientas de diseño de sistemas inteligentes. Proveen facilidades para diseñar bases de datos inteligentes. Aunque el campo del diseño de base de datos, diseño de sistemas de información y diseño de sistemas expertos tienen separado su mantenimiento en el pasado, su integración en una base de datos inteligente es esencial. Así, estas herramientas permiten a los desarrolladores y administradores de sistemas un mejor diseño y mantenimientos de la base de datos inteligente.

High-level user interface.

Este nivel crea el modelo de la tarea y ambiente de base de datos con el que el usuario interactuará.

La interfaz del usuario es presentada en dos aspectos:

- Éste es un modelo central que es presentado al usuario, consiste de la representación orientada a objetos de la información con una colección de herramientas integradas para crear nuevos tipos de objetos, buscando y respondiendo preguntas.
- Son una colección de herramientas de alto nivel, el cual aunque no es una parte esencial del modelo central, da un realce a la funcionalidad del sistema de base de datos inteligente para ciertas clases y usuarios.

Normalmente se pueden distinguir dos niveles de la interfaz del usuario, las cuales son:

- El nivel físico. es típicamente más obvio, consistiendo de entradas y salida de dispositivos, semejantes al ratón, teclado, monitor
- El nivel cognitivo. es más difícil para describir, consistiendo del modelo subyacente usado para presentar la información, la interpretación que el usuario entonces hace y las intensiones que entonces el usuario formula.

Intelligent database engine.

Este es el nivel base. El motor de base de datos inteligente incorpora un modelo que permite hacer una deductiva orientación a objetos representación de la información, que puede ser expresada y operada sobre una variedad de caminos. El motor incluye procedimientos de inferencia de encadenamiento hacia delante y hacia atrás. Muchas de estas características del motor integrado dependerán sobre las especificaciones del ambiente de hardware y software en el cual la base de datos inteligente es implementada.

La interfaz del usuario es soportada por una colección de capacidades de la base de datos. Estas capacidades son el mecanismo que permite un sistema de administración de base de datos o aplicaciones de administración de información su funcionamiento. Ejemplo de estas capacidades incluye el procesamiento de query's y la habilidad para llevar fuera del razonamiento deductivo.

La inteligencia de la interfaz del usuario esta en gran parte determinada por la inteligencia de la aplicación subyacente. Las siguientes son algunas características de un sistema de base de datos y este nivel que realza el total de inteligencia del sistema.

- Modelo de datos basado en el conocimiento y orientado a objetos. Permite la representación de información en una forma la cuál refleja lo más fácilmente posible la percepción de los usuarios del mundo real.
- Base de datos integrada y motor de inferencia. Siguen después del uso del modelo de datos basado en el conocimiento. Éstos permiten la recuperación deductiva para ser llevado en un ambiente fuera donde la búsqueda e inferencia de la información son integradas
- Búsquedas sensitivas al contexto y/o de estructura sensitiva. Envuelve la recuperación del conocimiento basado sobre la forma. La búsqueda sensitiva envuelve sabiendo donde buscar la información relevante basado en el contexto
- Soporte de múltiples medios de almacenamiento. Permiten una variedad de tipos (gráficas, sonido, etc.) para ser eficientemente almacenados y recuperados dentro de la base de datos
- Administración inteligente de versión, recuperación y resistencia. Asegura que las versiones previas y la actual de las bases de datos de desarrollo, sean recuperadas eficientemente. La recuperación y resistencia son las ediciones que tratan el grado a el cual la base de datos maneja las demandas operacionales que experimenta, incluyendo simulación de acceso de muchos usuarios y fallas de equipo
- Soporte de transacciones y concurrencia. Así como muchas convencionales bases de datos, las bases de datos inteligentes soportan transacciones atómicas concurrentes
- Optimización de query's. En adición, el motor subyacente de una base de datos inteligente ejecuta extensiva optimización de query para proveer adecuadas respuestas de tiempo real para complejos queries envolviendo bases de conocimiento orientadas a objetos

Herramientas CASE

Desde el inicio de la creación del 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.

El significado de las siglas CASE viene de su acrónimo en ingles Computer Aided Software Engineering (Ingeniería Asistida por Computadora).

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.
No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase en común. Podrían clasificarse así:
- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que abarca.
- La arquitectura de las aplicaciones que produce.

Las herramientas CASE, en función de las fases del ciclo de vida que cubre, se pueden agrupar de la forma siguiente:
- Herramientas integradas, I-CASE (Integrated CASE): Abarcan todas las fases del ciclo de vida del desarrollo de sistemas son llamadas CASE workbench.
- 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.
- Herramientas de bajo nivel, L-CASE (Lower CASE): Dirigidas a las últimas fases del desarrollo, construcción e implantación.
- Juegos de herramientas, (Tools 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.

Componentes de una herramienta CASE

- Repositorio: Éste se puede definir como la base de datos central de una herramienta CASE. El repositorio amplía el concepto de diccionario de datos para incluir toda la información que se va generando a lo largo del ciclo de vida del sistema, por ejemplo: componentes de análisis y diseño.
- Módulos de diagramación y modelación: Algunos de los diagramas y modelos utilizados con mayor frecuencia son: diagrama de flujo de datos, modelo E-R y técnicas matriciales.
- Herramienta de prototipado: El objetivo principal de esta herramienta es poder mostrar al usuario, desde los momentos iniciales del diseño, el aspecto que tendrá la aplicación una vez desarrollada. Ello facilitará la aplicación de los cambios que se consideren necesarios, todavía en la fase de diseño.
- Generador de código: Normalmente se suele utilizar sobre ordenadores personales o estaciones de trabajo, por lo que el paso posterior del código al host puede traer problemas, al tener que compilar en ambos entornos.
- Módulo generador de documentación: El módulo generador de la documentación se alimenta del repositorio para transcribir las especificaciones allí contenidas.

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.
- 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 está integrado con Oracle Developer, lo cual provee una solución para desarrollar sistemas empresariales de segunda generación.

- Ø System Architect: Herramienta que posee un repositorio único que integra todas las herramientas y metodologías usadas. En la elaboración de los diagramas, el System Architect conecta directamente al diccionario de datos, los elementos asociados, comentarios, reglas de validaciones, normalización.
- Ø 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.
- Ø TableDesigner: Herramienta que permite la creación de bases de datos en Access y SQL Server de Microsoft.

Conclusiones

La explosión tecnológica ha permitido desarrollar productos más robustos, potentes, complejos y de mayor funcionalidad. Aunado a lo anterior, se tiene la competencia comercial que desarrolla nuevos productos o nuevas versiones de los viejos con nuevas funcionalidades.

Este crecimiento en tecnología requiere que el conocimiento necesario para dar soporte al área de sistemas sea tan grande que se tiene que seccionar en especializaciones.

Las actividades del área de sistemas ya no se pueden llevar por una sola persona sino que se ha distribuido en varios profesionistas. Por ejemplo, se requiere personal que atienda las configuraciones de las estaciones de trabajo, otros que configuren los servidores y sus aplicaciones, otras más que vigilen el comportamiento de la red de comunicación, etc.

Es por esto que el administrador del área de sistemas no tiene la obligación de dominar completamente todas las actividades necesarias, lo que si es su obligación es conocer al menos lo básico que le permita comunicarse con otros profesionistas que cubran sus deficiencias.

Es muy importante que el administrador de sistemas tenga un grado de conocimiento general aceptable por lo que requiere de capacitación extensa y continua.

La formación que da la Universidad Autónoma de México a los ingenieros permite que sean capaces de desarrollarse en cualquier actividad del área de sistemas y que esto pueda ser el inicio de una carrera que le permita llegar a ser el Administrador del área de Sistemas.

Como se mencionó anteriormente la velocidad con la que cambia la tecnología es demasiado rápida, por lo que seguramente ya existen nuevas versiones de los productos tratados en este trabajo.

Sin embargo, una de las características de los productos de cómputo es que sin importar la marca o versión todos tienen semejanzas. Es decir, lo que aprendes en un sistema operativo te puede ser útil para otro.

Unos consejos a mis incipientes colegas son que se profundicen en sus conocimientos actuales de informática, que traten de aprender nuevas actividades y que se mantengan actualizados de los avances de la tecnología

Bibliografía

Análisis Y Diseño De Sistemas
Kendall& Kendall

Análisis y diseño orientado a objetos,
Booch, Grady
México, Addison-Wesley

Aplique SQL
Groff, James R. y Weinberg Paul N.Osborne
McGraw-Hill

The Art Of Unix Programming,
Eric Steven, Raymond

Analysis and System Design. Developing. Information Systems with UML.
Maciaszek, L.A..
Addison-Wesley

CISA Review Manual
México, Trillas

COBIT, Control Objetives Management Guidelines Maturity Models

El Lenguaje Unificado de Modelado. Manual de Referencia
Booch, Grady.
España, Addison-Wesley Iberoamericana.

El Proceso Unificado de Desarrollo de Software
Jacobson, Ivar.
España, Addison-Wesley Iberoamericana.

Introduction HOW-TO BASH Programming
Mike G Mikkey

Introducción a los sistemas de bases de datos
Date, Chris.

Introducing Microsoft® Windows® Server
Jerry Honeycutt

Introducción a la administración de sistemas
RedHat (Red Hat Enterprise Linux

Learning Red Hat Linux
Bill McCarty
O'Reilly

Learnig the Vi Editor
O'reilly

Learning Windows Server 2003
OReilly

Learning the bash Shell
O'Reilly

Linux: Device Drivers
Jonathan Corbet, Greg Kroah-Hartman, Alessandro Rubini
O'Reilly

Maximun RPM - Taking the RPM Package Manager to the Limit"

Edward C. Bailey.
Red Hat

Organización de las Bases de Datos
Martin, James.
Prentice-Hall Hispanoamericana

Operating System Desing and Implementation,
Andrew S. Tanenbaum,
Prentice Hall

Object oriented software engineering.
Jacobson, Ivar.
EE.UU, Addison-Wesley

Object oriented modeling and design.
Rumbaugh, James y otros.
EE.UU, Prentice-Hall

Reingeniería de la Auditoría Informática
SOLIS Montes Gustavo Adolfo.,
México, Trillas, México

UML Gota a Gota
Fowler, Martin
México, Addison-Wesley

Mesografía

ASP
http://www.auladigital.com/aula.php3?action=category&category=104
http://www.asptutor.com/asp/default.asp
http://www.soloasp.com.ar/index.asp
http://www.programacion.com/asp/tutorial.asp_basics.html

Cetus Links - Object-Orientation
http://www.cetus-links.org

Filesystem Hierarchy Standard
http://www.pathname.com/fhs/

BASH Programming - Introduction HOW-TO
http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
http://www.insflug.org/COMOs/Bash-Prog-Intro-COM

Java Server Pages
http://mipagina.cantv.net/williamyanez/jsp/default.htm
http://java.sun.com/products/jdk/1.1/docs/guide/jdbc/getstart/introTOC.doc.
http://www.programacion.com/java/tutorial.php?id=servlets_jsp
http://www.javahispano.org/tutoriales/jsp/

PHP
http://www.php.net
http://geneura.ugr.es/~maribel/php/

Objetct Management Group
www.service-architecture.com/object-oriented-databases
http://www.omg.org/technology/uml/index.htm
ttp://www.odbms.org/

Object Orientation
http://ootips.org/
http://www.well.com/user/ritchie/oo.html

UML Rational Software Corporation
http://www.rational.com/uml

Sinan Si Alhir's Web Site
http://home.earthlink.net/~salhir/