UNIVERSIDAD SALESIANA DE BOLIVIA
CONTADURÍA PÚBLICA

DOSSIER
GESTIÓN I – 2014

BASE DE DATOS I
CUARTO SEMESTRE
Paralelos: 4A1
4A2

Ing. Claudia Teresa Llanos Vidaurre

INDICE

I INTRODUCCION.

Los sistemas de información orientada a base de datos brinda un soporte tan eficiente como para la toma de decisiones, control de transacciones, y almacenamiento lógica de datos, con la visión de este contexto, la materia trata de brindar herramientas que puedan ayudar a realizar este tipo de enfoques, proporcionando conceptos teóricos-prácticos de tal forma que el alumno este inmerso en el campo de trabajo que nos brinda el mercado globalizado

La contribución de la asignatura de BASE DE DATOS I hacia el logro de las competencias del perfil del profesional de la Carrera de Contaduría Pública y de Sistemas se da en los siguientes aspectos: *El profesional podrá fundamentar, analizar y diseñar un cúmulo de datos en base al modelamiento de base de datos, y manejo de gestores de base de datos*

El objeto de estudio tiene relación con el gran campo del conocimiento relacionado con la Programación de Sistemas y las Ciencias de la Computación.

- **Competencias generales educativas:**

  - Resuelve problemas a través del diseño de base de datos.
  - Valora los programas de software de gestión de base de datos (SGBD)

- **Competencias generales instructivas:**

  - Fundamenta los conocimientos teórico-prácticos sobre modelamiento de base de datos
  - Diseña base de datos e-r como relacionales de manera correcta y eficientes para una variedad de sistemas inmersos en el área de la contabilidad y auditoria.

II OBJETIVOS DE LA MATERIA

- GENERAL

  El objetivo de la materia es el de dotarle capacidades de abstracción como para poder diseñar en los distintos modelos de datos cualquier situación de la vida realidad en especial aquellos inmersos dentro de los procesos contables

- ESPECÍFICOS

- Dar al estudiante una base en cuanto a manejo de un Sistema de Gestión de Base de Datos relacional
- Diseñar en los modelos E-R y modelo relacional
- Monitorear base de datos a través de Lenguajes de Consulta

- ADICIONAL

Implementar el Estilo Salesiano en el proceso enseñanza aprendizaje, enfatizando en los pilares básicos: RAZÓN, AMOR Y RELIGIÓN

II. CONTENIDO O CUERPO DEL DOSSIER

SISTEMAS DE ARCHIVOS

Objetivos

Tratar las estructuras y principales características de los sistemas de archivos.

Files

- Field (campo)
- Record (registro)
- Field (archivo)
- Database (Base de Datos)

- ***Un campo*** (Field) es el elemento de datos básico. Un campo individual contiene un valor único. Esta caracterizado por su longitud y por el tipo de datos. Dependiendo del diseño del archivo, los campos pueden ser de tamaño fijo o variable. Un campo pueden contener un subcampo.

- ***Registro (Record)*** es una colección de campos relacionados que pueden tratarse como una única unidad por un programa de aplicación. Por ejemplo:**,** un registro de empleados va contener campos como nombre, numero de seguridad social, etc.
  También dependiendo del diseño, los registros pueden ser de longitud fija o de longitud variable. Un registro va a tener una longitud variable si algunos de los campos son de tamaños variables o si el numero de campos es variable. Cada campo tiene un nombre de campo.

  - ***Archivo (File)*** es una colección de registros similares. El archivo es tratado como una entidad individual por los usuarios y las aplicaciones y puede ser referenciada por el nombre. Los archivos tienen nombres únicos y pueden crearse y borrarse. En un sistema compartido, los usuarios y los programas tienen garantizado o denegado el acceso a archivos completos. En algunos sistemas más complejos, dicho control se aplica a los registros o a los campos.

  - ***Base de datos (database)*** es una colección de datos relacionados. El aspecto esencial de la base de datos es que la relación que existe entre los elementos de datos es explícita y la base de datos es diseñada para usarse en un numero diferente de aplicaciones. Una base de datos puede contener toda la información relacionado a una organización o proyecto, corno un estudio de mercado o científico. La base de datos consiste en uno o más tipos de archivos.

Los usuarios y aplicaciones desean usar los archivos. Las operaciones típicas que deben soportarse incluyen las siguientes:

- **Recuperar Todo (Retrieve_all):** Recuperar todos los registros de un archivo. Esto va a requerir de una aplicación que deba procesar toda la información de un archivo una vez.. Esta opcion es usualmente equivalente con el término de sequential proccessing, (proceso secuencial), porque todos los registros son accedidos en secuencia.

- **Recuperar_Uno (Retrieve_One):** Esta operación requiere la recuperación de un solo un registro. Las soluciones interactivas orientadas a la transacción necesitan esta operación.

- **Recuperar_siguiente (Retrieve_Next):** Esta operación implica la recuperación del registro que es el siguiente, según una secuencia lógica, el recuperado hace menos tiempo. Un programa que realice búsquedas puede usar también esta operación.

- **Recuperar Previo (Retrieve_Previous):** Es similar a Recuperar Siguiente, pero en este caso el registro que es "previo" al que se esta accediendo en el momento actual.

- **Insertar Uno (Insert One):** Inserta un nuevo registro dentro del archivo. Es necesario que el nuevo registro se ajuste a una posición particular para preservar la secuencia del archivo.

- **Borrar uno (Delete One):** Borra un registro existente. Ciertos enlaces o otras estructuras puede que necesiten actualizarse para preservar la secuencia del archivo.

- **Actualizar Uno (Update_one):** Recupera un registro o actualiza uno o más de sus campos, y rescribe la actualización en el archivo. Es necesario preservar la secuencia con esta operación. Sí el tamaño del registro esta cambiado, la operación de actualización es más difícil si el tamaño es preservado.

- **Recuperar Varios (Retrieve_ Few):** Recupero un número de registros.

La naturaleza de las operaciones que comúnmente se ejecutan sobre un archivo va a influenciar sobre el modo en que se va a organizar el mismo.

Sistemas de Gestión de Archivos (File Management Sytems)

Un sistema de gestión de archivos es aquel sistema software que provee servicios a los usuarios y aplicaciones en el uso de archivos. El único camino que tiene el usuario o la aplicación tiene para acceder a los archivos es a través de un sistema de gestión de archivos. Esto revela para el usuario o programador la necesidad de desarrollar software de propósito especial para cada aplicación y provee al sistema un medio de controlar su ventaja más importante.
Estos son los objetivos de un sistema de gestión de archivos:

- Cumplir con las necesidades de gestión de datos y con los requisitos del usuario, que incluye el almacenamiento de, datos y la capacidad de ejecutar las operaciones en la lista precedente.
- Garantizar, en la medida de lo posible, que el dato en el archivo es valido.
- Optimizar el rendimiento, ambos desde el punto de vista del sistema en términos de productividad global, y como punto de vista del usuario en tiempos de respuesta.
- Para proveer soporte de E/S para una variedad de tipos de dispositivos de almacenamiento.
- Para minimizar o eliminar la posibilidad de perdida o destrucción de datos.
- Para proveer un conjunto estándar de rutinas de E/S.
- Para proveer soporte de E/S para múltiples usuarios, en caso de sistemas multiusuarios.

Arquitectura de los sistemas de Archivos **(File System Architecture)**

Un camino para hacerse una idea del alcance de la gestión de archivos es de mirar una representación típica de la organización del software, como se muestra en la figura de abajo:

Diferentes sistemas van a tener diferente organizaciones pero estas organizaciones son razonablemente representativas. A un nivel mas bajo los manejadores de dispositivos (device drivers) se comunican directamente con los dispositivos de periféricos o con sus canales o controladores. Un controlador de dispositivos es responsable de iniciar las operaciones de E/S en un dispositivo y procesar la terminación de una petición de E/S. Para operaciones de archivos, el controlador típico de dispositivos son discos y unidades de cinta. Los manejadores de los dispositivos son usualmente considerados como parte del sistema operativo.

El próximo nivel esta referido con el nombre de sistema de archivos básicos (basic file system), o nivel de E/S física (physical I/O) . Esta es la interfase primaria con el ambiente fuera del sistema de la computadora. Este nivel trata con bloques de datos que son intercambiados con sistemas de disco o cinta. De este modo. Se preocupa de ubicar dichos bloques en el dispositivo de almacenamiento secundario y del almacenamiento intermedio de los mismos en memoria principal. Este nivel no comprenderá el contenido de los datos o la estructura de los archivos implicados. El sistema de archivos básicos es usualmente considerado como parte del sistema operativo.

El supervisor básico de E/S (Basic I/O supervisor) es el responsable de la iniciación y terminación de todas las E/S con archivos. En este nivel, hay unas estructuras de control que se encargan de la entrada y de salida con los dispositivos la planificación y el estado de los archivos. El supervisor básico de E/S se encarga de seleccionar el dispositivo donde se va a realizar la E/S con los archivos dependiendo del archivo seleccionado. También se encarga de la planificación de los accesos a disco y cinta para optimizar el rendimiento. En este nivel se asignan los buffers de E/S y se reserva la memoria secundaria. El supervisor básico de E/S es parte del sistema operativo.

La E/S lógica habilita a los usuarios y aplicaciones de acceder a registros. Así mientras el sistema de archivos básico trabaja con bloques de datos. El modulo lógico de E/S trabaja con el archivo de registros. La E/S lógica provee una capacidad de E/S de registro de propósito general y mantiene los datos básicos acerca de los archivos.

El nivel del sistema de archivo más cercano de usuario es usualmente el método de acceso (access method). Provee una interfase estándar entre aplicaciones y los archivos del sistema a dispositivos que guarden datos. Los diferentes métodos de acceso reflejan las diferentes estructuras de datos y diferentes maneras de acceder y procesar el dato.

Funciones de la gestión de archivos (File management Functions)

Los usuarios y las aplicaciones interactúan con el sistema de archivos mediante comandos para crear y borrar archivos y realizar operaciones sobre los archivos. Antes de ejecutar alguna operación, los archivos del sistema deben identificar y localizar el archivo seleccionado. Esto requiere el uso de alguna clase de directorio que es reservado para describir la localización de todos los archivos, más sus atributos. Además, la mayoría de los sistemas compartidos aplican algún control de acceso a los usuarios: solamente los usuarios autorizados están permitidos para acceder a archivos particulares en determinados lugares. Las operaciones básicas que el usuario o el programa pueden ejecutar sobre un archivo se pueden realizar a nivel de registro. El usuario o la aplicación ven el archivo con una estructura que organiza los registros, como una estructura secuencial. De este modo, para traducir las ordenes del usuario a ordenes

específicas de manipulación de archivos., debe emplearse el método de acceso apropiado para esta estructura de archivo.

Organización y acceso a archivos (File organizittion and access)

En esta parte vamos a usar el término organización de archivos para referirnos a la estructura lógica de los registros determinada por la manera en que se accede a ellos. La organización fisica del archivo en almacenamiento secundario depende de la estrategia de agrupación y de la estrategia de asignación de archivos.

Para seleccionar una organización de archivos hay diversos criterios que son importantes:

- Acceso Rápido para recuperar la información
- Fácil actualización
- Economía de almacenamiento
- Mantenimiento simple.
- Fiabilidad para asegurar la confianza de los datos.

La prioridad relativa de estos criterios va a depender de las aplicaciones que va a usar el archivo.
El número de alternativas de organización de archivos que se han implementado o propuesto es inmanejable, incluso para un libro dedicado a los sistemas de archivos.
La mayor parte de las estructuras empleadas en los sistemas reales se encuadran en una de estas categorías o puede implementarse como una combinación de estas:

- Pilas (The pile)
- Archivos secuenciales (sequential file)
- Archivos Secuenciales indexados. (indexed sequential file)
- Archivos indexados.(indexed file)
- Archivos directos o de dispersión (direct, or hashed, file).

Pilas

La forma menos complicada de organización de archivos puede denominarse la pila. Los datos se recolectan en el orden en que llegan. Cada registro consiste en una ráfaga de datos. El propósito de la pila es simplemente acumular la masa de datos y guardarlo.
Como no hay estructura para el archivo de la pila. el acceso a registro es por búsqueda exhaustiva..Si se quiere todos los registros que contienen un campo particular o que tienen un valor determinado para ese campo, debe buscarse en el archivo entero.
Los archivos de pilas se aplican cuando los datos se recogen y almacenan antes de procesarlos o cuando no son fáciles de organizar. Este tipo de archivo usa bien el espacio cuando los datos almacenados varían en tamaño y en estructuras. Este tipo de archivos no se adapta a la mayoría de las aplicaciones.

Archivos Secuenciales

La forma mas común de estructura de archivo es el archivo secuencial. En este tipo de archivo, un formato fijo es usado para los registros. Todos los registros tienen el mismo tamaño, constan del mismo número de campos de tamaño fijo en un orden particular. Como se conocen la longitud y la posición de cada campo, solamente los valores de los campos se necesitan almacenarse; el nombre del campo y longitud de cada campo son atributos de *la* estructura de archivos.

Un campo particular, generalmente el primero de cada registro se conoce como el campo clave. El campo clave identifica unívocamente al registro. Así, los valores de la clave para registros diferentes son siempre diferentes.
Los archivos secuenciales son típicamente utilizados en aplicaciones de proceso de lotes Y son óptimos para dichas aplicaciones si se procesan todos los registros. La organización secuencias de archivos es la única que es fácil de usar tanto en disco como en cinta.
Para las aplicaciones interactivas que incluyen peticione s o actualizaciones de registros individuales, los archivos secuenciales ofrecen un rendimiento pobre.
Normalmente un archivo secuencial se almacena en bloques, en un orden secuencial simple de los registros. La organización física del archivo en una cinta o disco se corresponde exactamente con la ubicación lógica del archivo. En este caso, el procedimiento para ubicar los nuevos registros en un archivo de pila separado, llamado archivo de registro (log file) o archivo de transacciones. Periódicamente, se realiza una actualización por lotes que mezcla el archivo de registro con el archivo maestro para producir un nuevo archivo en secuencia correcta de claves.

Archivos Secuenciales indexados

Un método popular para superar las desventajas de los archivos secuenciales es el del archivo secuencias indexado. El archivo secuencial indexado mantiene las características básicas de los archivos secuenciales: los registros están organizados en una secuencia basada en un campo. Dos características se añaden: un índice del archivo para soportar los accesos aleatorios y un archivo de desbordamiento (overflow). El índice provee una capacidad de búsqueda para llegar rápidamente a las proximidades de un registro deseado. El archivo de desbordamiento (overflow) es similar al archivo de registro usado en un archivo secuencial, pero esta integrado de forma que los registros del archivo de desbordamiento se ubican en la dirección de un puntero desde si registró precedente. En la estructura secuencial indexada más simple, se usa un solo nivel de indexación. El índice, en este caso, es un archivo secuencial simple. Cada registro del archivo índice tiene dos campos: un campo clave, que es el mismo que el campo clave del archivo principal y un puntero al archivo principal. Para encontrar un campo específico se busca en el índice hasta encontrar el valor mayor de la clave que es igual o precede al valor deseado de la clave. La búsqueda continúa en el archivo principal a partir de la posición indicada por el puntero.

Archivos Indexados

Los archivos secuenciales indexados retienen la limitación del archivo secuencial: la eficacia en el procesamiento se limita al basado en un único campo del archivo. Cuando es necesario buscar un registro basándose en algún otro atributo distinto del campo clave ambas formas de archivo secuencial no son adecuadas. En algunas aplicaciones esta flexibilidad es deseable.
Para alcanzar esta flexibilidad, se necesita una estructura que utilice múltiples índices, uno para cada tipo de campo que pueda ser objeto de la búsqueda.
Se suelen utilizar dos tipos de índices. Uno índice exhaustivo contiene una entrada par cada registro del archivo principal. Otro índice parcial contendrá entradas a los registros donde este el campo de interés. Con registros de longitud variable, algunos registros no contendrán todos los campos.
Los archivos indexados son muy utilizados en aplicaciones donde es crítica la oportunidad de la información y donde los datos son rara vez procesados de forma exhaustiva.

Archivos Directos o de Dispersión

Los archivos directos explotan la capacidad de los discos para acceder directamente a cualquier bloque de dirección conocida. Como en los archivos secuenciales y secuenciales indexados, se requiere un campo clave en cada registro. Sin embargo, aquí no hay concepto de ordenamiento secuencial.

Directorios de Archivo

Asociado con algunos sistemas de gestión de archivos o cualquier colección de archivos suele haber un directorio de archivos. El directorio contiene información acerca de los archivos, incluyendo atributos, localización y propietario. Mucha de esta información, especialmente la concernida con el almacenamiento es gestionada por el sistema operativo. El directorio es propiamente un archivo, poseído por el sistema operativo y, accesible a traces de diversas rutinas de gestión de archivos. Aunque alguna información en los directorios esta disponible para los usuarios y aplicaciones, en general , la información se proporciona indirectamente a través de rutinas del sistema. De este modo los usuarios pueden acceder directamente al directorio, incluso en modo de solo lectura.

Estructura

La manera en que la información se almacena difiere mucho en los diferentes sistemas. Parte de la información puede almacenarse en un registro de cabecera asociado al archivo, esto reduce el espacio necesario para el directorio, haciendo más fácil mantener todo el directorio.

La forma mas fácil de estructuración de un directorio es una lista de entradas, unas para cada archivo. Esta estructura puede representarse con un simple archivo secuencial, con el nombre del archivo haciendo las veces de clave.
Operaciones que se pueden realizar con un directorio:

- Buscar: Cuando alguien referencia el archivo, debe buscarse en el directorio la entrada correspondiente al archivo.
- Crear archivo: Al crear un nuevo archivo. debe añadirse una entrada al directorio.
- Borrar archivo: Al borrar un archivo, debe eliminarse una entrada al directorio.
- Listar directorio: Puede solicitarse todo el directorio o una parte.

Una simple lista no se ajusta bien a estas operaciones. Si el directorio es una simple lista secuencias, no ofrecerá ayuda en la organización de los archivos y obligara al usuario a tener cuidado de no usar el mismo nombre para dos tipos diferentes de archivos. Para resolver este problema se puede acudir a un esquema de dos niveles donde hay un directorio para cada usuario y un directorio maestro.
Un método más potente y flexible es el directorio jerárquico o estructurado en árbol. Existe un directorio maestro que contiene un número determinado de directorios de usuario. Cada uno de estos directorios puede tener a su vez subdirectorios y archivos como entradas. Esto se cumple en cualquier nivel.
Para organizar cada directorio y subdirectorio. El método más simple es almacenar cada directorio como un archivo secuencial. Cuando los directorios contengan un número muy grande de entradas, tal organización puede conducir a tiempos de, búsqueda innecesariamente grandes. En ese caso se prefiere una estructura de dispersión.

Designación (Naming)

Los usuarios necesitan poder referirse a un archivo mediante un nombre simbólico. Cada archivo del sistema debe tener un nombre único para que las referencias al archivo no sean ambiguas. Por otro lado, es una carga inaceptable para los usuarios el proporcionar nombres únicos, especialmente en los sistemas compartidos.
El uso de directorios estructurados en árbol minimiza la dificultad de asignar nombres únicos. Cualquier archivo del sistema puede ser localizado siguiendo un camino desde, el directorio raíz o maestro. Descendiendo por varias ramas hasta que se alcance el archivo. La serie de nombres de directorios, terminados con el propio nombre del archivo, constituye el propio nombre del camino del archivo.
Cada usuario interactivo o proceso tiene asociado un directorio actual, conocido a menudo como directorio de trabajo.

El Compartir Archivos (File Sharing)

En un sistema multiusuario, casi siempre existe la necesidad de permitir a los usuarios Compartir archivos. Dos problemas surgen:

- Los derechos de accesos
- Gestión de los accesos simultáneos

Derechos de Acceso:
El sistema de archivos provee una herramienta flexible para permitir compartir extensos archivos entre los usuarios. El sistema de archivos debe proporcionar un numero de opciones de modo en que un archivo que es accedido pueda ser controlado. Normalmente, al usuarios o a los grupos de usuarios se les otorgan ciertos derechos de acceso a cada archivo. Un amplio rango de derechos de acceso se ha venido usando. La siguiente lista representa los derecho de acceso que pueden ser asignados a un usuario en particular para un archivo en particular:

- Ninguno: El usuario no puede siquiera determinar la existencia del archivo ni mucho menos acceder al mismo. No se permite al usuario leer el directorio de usuario que incluya al archivo.
- Conocimiento: El usuario sabe de la existencia del archivo Y quien el dueño. El usuario puede solicitar los derechos de acceso adicionales al propietario.
- Ejecución: El usuario puede ejecutar y cargar un programa pero no copiarlo.
- Lectura: El usuario puede leer el archivo para cualquier propósito, incluyendo copia y ejecución.
- Adición: El usuario puede añadir datos al archivo, generalmente al final, pero no puede modificar o borrar el contenido del mismo.
- Actualización: El usuario puede modificar, borrar y añadir otros datos al archivo.
- Cambio de protección: El usuario puede cambiar los derechos de acceso otorgados a usuarios.
- Borrado: El usuario puede borrar el archivo del sistema de archivos.

Los derechos constituyen una jerarquía. Si un usuario adquiere el derecho de la actualización para un archivo determinado, también habrá adquirido los derechos siguientes: conocimiento, ejecución, lectura y adición.

El propietario de un archivo dispone de los derechos de acceso listados antes y puede otorgar derechos a los otros. Puede ofrecerse acceso a las siguientes clases de usuarios:

- Usuario específico: Usuarios individuales quienes son designados por su ID de usuario.
- Grupos de usuarios: Un conjunto de usuarios no definidos individualmente.
- Todos: Todos los usuarios que tengan acceso al sistema. Estos serán archivos públicos.

Accesos Simultáneos:
Cuando el acceso es concedido para añadir o actualizar un archivo a mas de un usuario, el sistema operativo o el sistema de gestión de archivos debe hacer cumplir una disciplina. Un método de fuerza bruta consiste en permitir a los usuarios bloquear el archivo entero cuando lo vaya a actualizar. Un mejor control es bloquear los registros individuales durante la actualización. Al disertar la posibilidad de accesos comparados, deben abordarse aspectos de exclusión mutua e interbloqueo.

Agrupación de Registros (Record Blocking)

Para realizar E/S, los registros deben organizarse en bloques. Dado un tamaño de bloque, pueden seguirse los siguientes tres métodos de agrupación en bloques:

- Bloques fijos: Se usan registros de longitud fija y un número entero de registros son Guardados en un bloque. Puede haber espacio sin usar al final de cada bloque.
- Bloque de longitud variable por tramos: Se usan registros de longitud variable y agrupada en bloques sin dejar espacios sin usar.
- Bloque de longitud variable sin tramos: Son usados registros de longitud variable, pero no se dividen en tramos. En la mayoría de los bloques habrá un espacio desperdiciado, debido a la imposibilidad de aprovechar el resto del bloque si el registro siguiente es mayor que el espacio sin usar restante. Los bloques de tamaño fijo son el modo más común de archivos secuenciales con registro de longitud variable. Los bloques de longitud variable por tramos constituyen un almacenamiento eficaz y no ponen límites al tamaño de los registros. Pero esta técnica es difícil de implementar.

Gestión del Almacenamiento secundario (Secondary Storage Management)

En el almacenamiento secundarlo, un archivo consiste en una colección de bloques. El sistema de gestiónele archivos es el responsable de la asignación de los bloques a archivos. Esto crea dos problemas sobre la gestión. Primero, el espacio en el almacenamiento secundario debe ser designado a los archivos, en segundo lugar, es necesario guardar constancia del espacio disponible para asignar. Estas dos tareas están relacionadas, el método tomado para asignar los archivos puede influir en el método de gestión del espacio libre. También existe una interacción entre la estructura de archivo y la política de asignación.

Asignación de Archivos

Surgen varias cuestiones:
1. Cuando se crea un archivo nuevo. ¿Se asigna de una sola vez el máximo espacio que necesite?
2. El espacio se asigna a un archivo en forma de una o más unidades contiguas que se llaman secciones. Un tamaño de una sección puede variar desde un único bloque a un archivo entera. Que tamaño de sección debería usarse para asignar archivos?
3. ¿Qué tipos de estructura de datos o tabla se usaran para guardar constancia de las secciones asignadas a un archivo. Dicha tabla se conoce normalmente como tabla de asignación de archivos (FAT).

Asignación previa frente a Asignación dinámica

Una política de asignación requiere que el tamaño máximo de un archivo sea declarado un el momento de crearlo. En un número de casos, como el compilar los programas, la producción del resumen de los datos del archivo, o la transferencia de un archivo desde otro sistema por una red de comunicaciones, este valor puede estimarse. Pero en muchas aplicaciones es difícil estimar el tamaño máximo del archivo.

Tamaño de Sección

La segunda cuestión de la lista anterior es la del tamaño de sección asignada a los archivos. En un extremo, se puede asignar una sección suficientemente grande para guardar el archivo entero. En el otro extremo, se asigna el espacio en disco de bloque en bloque. Al elegir el tamaño de sección, debe haber un compromiso relativo a la eficiencia desde el punto de vista de un solo archivo frente al del sistema global.

1- La contigüidad del espacio aumenta el rendimiento.
2- Disponer de un gran número de secciones pequeñas aumenta el tamaño de las tablas necesarias para gestionar la asignación de información.
3. Disponer de secciones de tamaño fijo simplifica la resignación del espacio.
4. Disponer de secciones de tamaño variable o secciones pequeñas de tamaño fijo minimiza la perdida de espacio no usado provocado por la sobre asignación.

- **Secciones contiguas variables y grandes:** Esta opción ofrecerá un rendimiento mejor. El tamaño variable evitara la perdida y las tablas de asignación de archivos serán pequeñas. El espacio es difícil de reutilizar.
- **Bloques:** Las secciones fijas y pequeñas ofrecen una flexibilidad mayor. La contigüidad se abandona y los bloques se asignan a medida que se necesitan.

Cualquier opción es compatible con la asignación previa o con la asignación dinámica. En el primer caso se asigna previamente a los archivos un grupo contiguo de bloques. En el segundo caso, todas las secciones necesarias son asignadas de una vez, Entonces la tabla de asignación de archivos permanecerá con tamaño fijo.

No esta claro que estrategia es la mejor. La dificultad de moldear estrategias alternativas esta en que intervienen muchos factores incluyendo los tipos de archivo, la pauta a los accesos a archivo, el grado de multiprogramación, etc.

Métodos de Asignación de Archivos

Con Asignación contigua: Cuando se crea un archivo se le asigna un unico conjunto contiguo de bloques. Esta es una estrategia de asignación previa que emplea secciones de tamaño variable. La tabla de asignación de archivos necesita solo una entrada por cada archivo, que muestre el bloque de comienzo y la longitud del archivo. La asignación contigua es la mejor desde el punto de vista de un archivo secuencias individual.

Con Asignación encadenada: La asignación normalmente se hace con bloques individuales. Cada bloque contendrá un puntero al siguiente bloque de la cadena. La tabla de asignación de archivos necesita de nuevo una sola entrada por cada archivo que muestre el bloque de comienzo y la longitud del archivo. No hay que preocuparse por la fragmentación externa porque solo se necesita un solo bloque cada vez. Este tipo de organización física se ajusta mejor a los archivos secuenciales que van a ser procesados secuencialmente.

La asignación indexada: Trata mucho de los problemas de las asignaciones contigua y encadenada. La tabla de asignación de archivos contienen un índice separado de un nivel para cada archivo; el índice posee una entrada para cada sección asignada al archivo. Los índices no están almacenados físicamente como parte de la tabla de asignación de archivos. El índice del archivo se guardara en un bloque aparte y la entrada del archivo en la tabla de asignación apuntada a dicho bloque. La asignación indexada soporta tanto el acceso secuencial como el acceso directo a los archivos.

Gestión del Espacio Libre

Al igual que al espacio asignado a los archivos, se debe gestionar el espacio que no queda asignado actualmente a ningún archivo. Para llevar a cabo cualquiera de las técnicas de asignación que se han descrito, es necesario saber que bloques del disco están disponibles. Hace falta una tabla de asignación de disco además de una tabla de asignación de archivos. Tres técnicas son de uso común:

- Las tablas de bits.
- Las secciones libres encadenadas.
- Y la indexación.

Tablas de Bíts

El método de las tablas de bits utiliza un vector que contiene un bit por cada bloque del disco. Cada entrada de igual a 0 corresponde a u bloque libre y cada 1 corresponde a un bloque en uso. Las tablas de bits tienen la ventaja de que es relativamente fácil encontrar un bloque o un grupo continuo de bloques libres. Las tablas de bits trabajan bien con cualquiera de los métodos de asignación de archivos. Otra ventaja es que puede ser tan pequeña como sea posible y puede mantenerse en memoria cada vez que se realice una asignación.

Secciones libres encadenadas

Las secciones libres pueden encadenarse juntas mediante un puntero y un valor de longitud en cada sección libre. Este método tiene un gasto mínimo porque no hay necesidad de tabla de asignación de disco, sin simplemente un puntero al comienzo de la cadena y la longitud de la primera sección. Este método sirve para todas las técnicas de asignación de archivos.

Indexación

El método de indexación trata el espacio libre como si fuera un archivo y utiliza una tabla índice. Por razones de eficiencia, el índice debe trabajar con secciones de tamaño variable mejor que con bloques. De este modo, habrá una entrada en la tabla para cada sección libre del disco. Este procedimiento ofrece un soporte eficaz para todos los métodos de asignación de archivos.

Fiabilidad

Considérese el escenario siguiente:

1. El usuario A solicita una asignación para añadir datos a un archivo existente.
2. La petición se atiende y se actualizan en memoria principal las tablas de asignación de disco y archivos, pero no aun en el disco.
3. El sistema se hunde y a continuación se reinicia
4. El usuario B solicita una asignación y se le otorga un espacio en el disco que se solapa con la ultima asignación hecha al usuario A.

5. El usuario A accede a la sección solapada mediante una referencia que esta almacenada en el archivo de A.

Esto surge debido al que el sistema mantiene copias de la tabla de asignación de disco y la tabla de asignación de archivos en memoria principal. Para evitar esto puede seguir los siguientes pasos:

1. bloquear en el disco la tabla de asignación de disco
2- Buscar espacio disponible en la tabla de asignación de disco.
3- Asignar el espacio, actualizar la tabla de asignación de disco y actualizar el disco.
4. Actualizar la tabla de asignación de archivos y actualizar el disco.
5. Desbloquear la tabla de asignación de disco.

Acronimos

UCP	Unidad Central de Proceso
S.O.	Sistema Operativo
E/S	Entrada y Salida
FAT	File allocation Table

Bibliografia

- Sistemas Operativos. William Stalling
- Notas sobre sistemas operativos. Carlos Neetzel

INTRODUCCIÓN A LOS CONCEPTOS DE BASES DE DATOS

1.1 Definición de Base de Datos

Todo buen curso necesita empezar con algunos conceptos básicos para el mejor entendimiento del mismo, por lo tanto empezaremos con las definiciones que involucran a las bases de datos.

Dato:
Conjunto de caracteres con algún significado, pueden ser numéricos, alfabéticos, o alfanuméricos.

Información:
Es un conjunto ordenado de datos los cuales son manejados según la necesidad del usuario, para que un conjunto de datos pueda ser procesado eficientemente y pueda dar lugar a información, primero se debe guardar lógicamente en archivos.

Conceptos básicos de archivos computacionales.

Campo:
Es la unidad más pequeña a la cual uno puede referirse en un programa. Desde el punto de vista del programador representa una característica de un individuo u objeto.

Registro:
Colección de campos de iguales o de diferentes tipos.

Archivo:
Colección de registros almacenados siguiendo una estructura homogénea.

Base de datos:
Es una colección de archivos interrelacionados, pueden ser creados con un DBMS. El contenido de una base de datos engloba a la información concerniente (almacenadas en archivos) de una organización, de tal manera que los datos estén disponibles para los usuarios, una finalidad de la base de datos es eliminar la redundancia o al menos minimizarla. Los tres componentes principales de un sistema de base de datos son el hardware, el software DBMS y los datos a manejar, así como el personal encargado del manejo del sistema.

Objetivos de la utilización de bases de datos.

Los objetivos principales de las de bases de datos es disminuir los siguientes aspectos:

Redundancia e inconsistencia de datos.
Puesto que los archivos que mantienen almacenada la información son creados por diferentes tipos de programas de aplicación existe la posibilidad de que si no se controla detalladamente el almacenamiento, se pueda originar un duplicado de información, es decir que la misma información sea más de una vez en un dispositivo de almacenamiento. Esto aumenta los costos de almacenamiento y acceso a los datos, además de que puede originar la inconsistencia de los datos - es decir diversas copias de un mismo dato no concuerdan entre si -, por ejemplo: que se actualiza la dirección de un cliente en un archivo y que en otros archivos permanezca la anterior.

Dificultad para tener acceso a los datos.
Un sistema de base de datos debe contemplar un entorno de datos que le facilite al usuario el manejo de los mismos. Supóngase un banco, y que uno de los gerentes necesita averiguar los nombres de todos los clientes que viven dentro del código postal 78733 de la ciudad. El gerente pide al departamento de procesamiento de datos que genere la lista correspondiente. Puesto que esta situación no fue prevista en el diseño del sistema, no existe ninguna aplicación de consulta que permita este tipo de solicitud, esto ocasiona una deficiencia del sistema.

Aislamiento de los datos.
Puesto que los datos están repartidos en varios archivos, y estos no pueden tener diferentes formatos, es difícil escribir nuevos programas de aplicación para obtener los datos apropiados.

Anomalías del acceso concurrente.
Para mejorar el funcionamiento global del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que múltiples usuarios actualicen los datos simultáneamente. En un entorno así la interacción de actualizaciones concurrentes puede dar por resultado datos inconsistentes. Para prevenir esta posibilidad debe mantenerse alguna forma de supervisión en el sistema.

Problemas de seguridad.
La información de toda empresa es importante, aunque unos datos lo son más que otros, por tal motivo se debe considerar el control de acceso a los mismos, no todos los usuarios pueden visualizar alguna información, por tal motivo para que un sistema de base de datos sea confiable debe mantener un grado de seguridad que garantice la autentificación y protección de los datos. En un banco por ejemplo, el personal de nóminas sólo necesita ver la parte de la base de datos que tiene información acerca de los distintos empleados del banco y no a otro tipo de información.

Problemas de integridad.
Los valores de datos almacenados en la base de datos deben satisfacer cierto tipo de restricciones de consistencia. Estas restricciones se hacen cumplir en el sistema añadiendo códigos apropiados en los diversos programas de aplicación.

Abstracción de la información.
Una base de datos es en esencia una colección de archivos relacionados entre sí, de la cual los usuarios pueden extraer información sin considerar las fronteras de los archivos.

Un objetivo importante de un sistema de base de datos es proporcionar a los usuarios una visión *abstracta* de los datos, es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos. Sin embargo para que el sistema sea manejable, los datos se deben extraer eficientemente.

Existen diferentes niveles de abstracción para simplificar la interacción de los usuarios con el sistema; Interno, conceptual y externo, específicamente el de almacenamiento físico, el del usuario y el del programador.

Nivel físico.
Es la representación del nivel más bajo de abstracción, en éste se describe en detalle la forma en como de almacenan los datos en los dispositivos de almacenamiento(por ejemplo, mediante señaladores o índices para el acceso aleatorio a los datos).

Nivel conceptual.
El siguiente nivel más alto de abstracción, describe que datos son almacenados realmente en la base de datos y las relaciones que existen entre los mismos, describe la base de datos completa en términos de su estructura de diseño. El nivel conceptual de abstracción lo usan los administradores de bases de datos, quienes deben decidir qué información se va a guardar en la base de datos.

Consta de las siguientes definiciones:

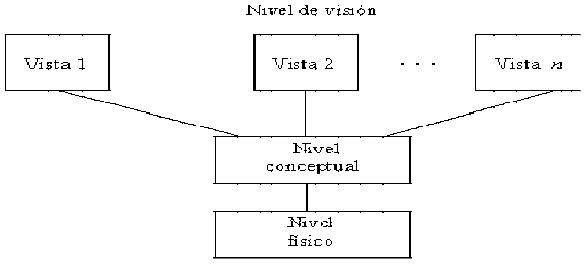
***Definición de los datos*:** Se describen el tipo de datos y la longitud de campo todos los elementos direccionables en la base. Los elementos por definir incluyen artículos elementales (atributos), totales de datos y registros conceptuales (entidades).

***Relaciones entre datos*:** Se definen las relaciones entre datos para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples.

En el nivel conceptual la base de datos aparece como una colección de registros lógicos, sin descriptores de almacenamiento. En realidad los archivos conceptuales no existen físicamente. La transformación de registros conceptuales a registros físicos para el almacenamiento se lleva a cabo por el sistema y es transparente al usuario.

Nivel de visión.
Nivel más alto de abstracción, es lo que el usuario final puede visualizar del sistema terminado, describe sólo una parte de la base de datos al usuario acreditado para verla. El sistema puede proporcionar muchas visiones para la misma base de datos.

La interrelación entre estos tres niveles de abstracción se ilustra en la siguiente figura.



Usuarios de las bases de datos.
Podemos definir a los usuarios como toda persona que tenga todo tipo de contacto con el sistema de base de datos desde que este se diseña, elabora, termina y se usa.

Los usuarios que accesan una base de datos pueden clasificarse como:

Programadores de aplicaciones.
Los profesionales en computación que interactuan con el sistema por medio de llamadas en DML (Lenguaje de Manipulación de Datos), las cuales están incorporadas en un programa escrito en un lenguaje de programación (Por ejemplo, COBOL, PL/I, Pascal, C, etc.)

Usuarios sofisticados.
Los usuarios sofisticados interactuan con el sistema sin escribir programas. En cambio escriben sus preguntas en un lenguaje de consultas de base de datos.

Usuarios especializados.
Algunos usuarios sofisticados escriben aplicaciones de base de datos especializadas que no encajan en el marco tradicional de procesamiento de datos.

Usuarios ingenuos.
Los usuarios no sofisticados interactuan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente en el sistema de base de datos, podemos mencionar al usuario ingenuo como el usuario final que utiliza el sistema de base de datos sin saber nada del diseño interno del mismo por ejemplo: un cajero.

Una **base** o **banco de datos** es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

En la actualidad, y gracias al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo en España, los datos personales se encuentran protegidos por la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD).

Tipos de bases de datos

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

Según la variabilidad de los datos almacenados

Bases de datos estáticas

Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

Según el contenido

Bases de datos bibliográficas

Solo contienen un surrogante (representante) de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resúmen o extracto de la publicación original, pero nunca el texto completo, porque sino estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias) [ver más abajo]

Bases de datos numéricas

Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

Bases de datos de texto completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

Directorios

Un ejemplo son las guías telefónicas en formato electrónico.

Texto de titular

Banco de imágenes, audio, video, multimedia, etc.==== Como su nombre lo indica, almacenan información en distintos formatos.

Bases de datos o "bibliotecas" de información Biológica

Son bases de datos que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

- Aquellas que almacenan secuencias de nucleótidos o proteínas.
- Las bases de datos de rutas metabólicas
- Bases de datos de estructura, comprende los registros de datos experimentales sobre estructuras 3D de biomoléculas
- Bases de datos clínicas
- Bases de datos bibliográficas (biológicas)

Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como *contenedor de datos* (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de *base de datos*; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

Bases de datos jerárquicas

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

Bases de datos de red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de *nodo*: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Base de datos relacional

Artículo principal: ***Modelo relacional***

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por *registros* (las filas de una tabla), que representarían las tuplas, y *campos* (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, *Structured Query Language* o *Lenguaje Estructurado de Consultas*, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años '80 (1980-1989) la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

Bases de datos orientadas a objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

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

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

Se está trabajando en **SQL3,** que es el estándar de SQL92 ampliado, que soportará los nuevos conceptos orientados a objetos y mantendría compatibilidad con SQL92.

Bases de datos documentales

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes. Tesaurus es un sistema de índices optimizado para este tipo de bases de datos.

Base de datos deductivos

Un sistema de **base de datos deductivas**, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática.

Gestión de bases de datos distribuida

La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades, sucursales de tiendas, etc.

CUESTIONARIO:

1. ¿Cuál es la diferencia entre una base de datos y un archivo?

_____
_____
_____
_____
_____
_____

2. ¿Cuáles son las razones para usar una base de datos?

_____
_____
_____
_____
_____
_____

3. ¿Cuáles son las diferencias entre el nivel físico y conceptual?

_____
_____
_____
_____
_____
_____

SISTEMA GESTOR DE BASE DE DATOS

Los **Sistemas Gestores de Bases de Datos** son un tipo de software muy específico, dedicado a servir de interfaz entre la Base de datos y el usuario, las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y *DataBase Management System*, su expresión inglesa.

Objetivos

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los usuarios de los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios *niveles de abstracción*.

- **Independencia**. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Redundancia mínima**. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

- **Consistencia**. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- **Integridad**. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- **Respaldo y recuperación**. Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Control de la concurrencia**. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

- **Tiempo de respuesta**. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Ventajas.

- Facilidad de manejo de grandes volumes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- Integridad referencial el terminar los registros.

Desventajas.

- El costo de actualización del hardware y software son muy elevados.
- Costo (salario) del administrador de la base de datos es costoso.
- El mal diseño de esta puede originar problemas a futuro.
- Un mal adiestramiento a los usuarios puede originar problemas a futuro.
- Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
- Generan campos vacíos en exceso.
- El mal diseño de seguridad genera problemas en esta.

SGBD libres

- PostgreSQL (http://www.postgresql.org Postgresql) Licencia BSD
- MySQL Licencia Dual, depende el uso.
- Firebird basada en la versión 6 de Interbase, Initial Developer's PUBLIC LICENSE Version 1.0.
- SQLite (http://www.sqlite.org SQLite) Licencia Dominio Público
- Sybase ASE Express Edition para Linux (Edición gratuita para Linux)

SGBD comerciales

- dBase
- Fox Pro
- IBM DB2 Universal Database (DB2 UDB)
- IBM Informix
- MAGIC
- Microsoft Access
- Microsoft SQL Server
- MySQL
- Oracle
- Paradox
- PervasiveSQL
- Progress (DBMS)
- Sybase ASE
- Sybase ASA
- Sybase IQ

MODELO ENTIDAD RELACION

El modelo E-R se basa en una percepción del mundo real, la cual esta formada por objetos básicos llamados entidades y las relaciones entre estos objetos así como las características de estos objetos llamados atributos.

Entidades y conjunto de entidades

Una ***entidad*** es un objeto que existe y se distingue de otros objetos de acuerdo a sus características llamadas atributos. Las entidades pueden ser concretas como una persona o abstractas como una fecha.

Un ***conjunto de entidades*** es un grupo de entidades del mismo tipo. Por ejemplo el conjunto de entidades CUENTA, podría representar al conjunto de cuentas de un banco X, o ALUMNO representa a un conjunto de entidades de todos los alumnos que existen en una institución.

Una entidad se caracteriza y distingue de otra por los ***atributos***, en ocasiones llamadas propiedades, que representan las características de una entidad. Los atributos de una entidad pueden tomar un conjunto de valores permitidos al que se le conoce como **dominio** del atributo. Así cada entidad se describe por medio de un conjunto de parejas formadas por el atributo y el valor de dato. Habrá una pareja para cada atributo del conjunto de entidades.

Ejemplo:
Hacer una descripción en pareja para la entidad alumno con los atributos *No_control, Nombre y Especialidad.*

Nombre_atributo, Valor

No_control , 96310418

Nombre , Sánchez Osuna Ana

Esp , LI

O considerando el ejemplo del Vendedor cuyos aributos son: RFC, Nombre, Salario.

Nombre_atributo, Valor

RFC , COMD741101YHR

Nombre , Daniel Colín Morales

Salario , 3000

Relaciones y conjunto de relaciones.

Una ***relación*** es la asociación que existe entre dos a más entidades.

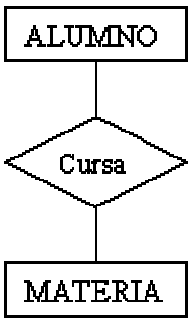
Un ***conjunto de relaciones*** es un grupo de relaciones del mismo tipo.

La cantidad de entidades en una relación determina el ***grado*** de la relación, por ejemplo la relación ALUMNO-MATERIA es de grado 2, ya que intervienen la entidad ALUMNO y la entidad MATERIA, la relación PADRES, puede ser de grado 3, ya que involucra las entidades PADRE, MADRE e HIJO.

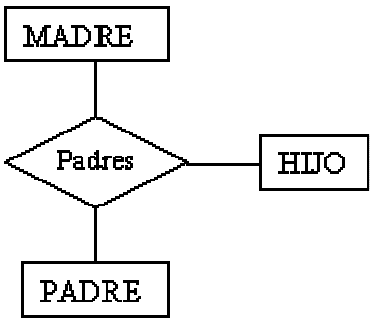
Aunque el modelo E-R permite relaciones de cualquier grado, la mayoría de las aplicaciones del modelo sólo consideran relaciones del grado 2. Cuando son de tal tipo, se denominan relaciones binarias.

La función que tiene una relación se llama ***papel***, generalmente no se especifican los papeles o roles, a menos que se quiera aclarar el significado de una relación.

Diagrama E-R (sin considerar los atributos, sólo las entidades) para los modelos ejemplificados:



Ejemplo de relación de grado 2



Ejemplo de relación de grado 3

Limitantes de mapeo.

Existen 4 tipos de ***relaciones*** que pueden establecerse entre entidades, las cuales establecen con cuantas entidades de tipo B se pueden relacionar una entidad de tipo A:

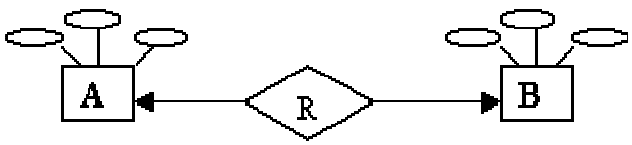
Tipos de relaciones:

- ***Relación uno a uno.***

Se presenta cuando existe una relación como su nombre lo indica uno a uno, denominado también relación de matrimonio. Una entidad del tipo A solo se puede relacionar con una entidad del tipo B, y viceversa;

Por ejemplo: la relación *asignación de automóvil* que contiene a las entidades EMPLEADO, AUTO, es una relación 1 a 1, ya que asocia a un empleado con un único automóvil por lo tanto ningún empleado posee más de un automóvil asignado, y ningún vehículo se asigna a más de un trabajador.

Es representado gráficamente de la siguiente manera:



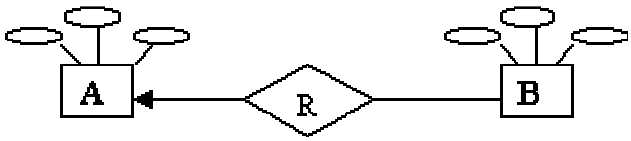
A: Representa a una entidad de cualquier tipo diferente a una entidad **B**.

R: en el diagrama representa a la relación que existe entre las entidades. El extremo de la flecha que se encuentra punteada indica el uno de la relación, en este caso, una entidad A ligada a una entidad B.

- ***Relación uno a muchos.***

Significa que una entidad del tipo A puede relacionarse con cualquier cantidad de entidades del tipo B, y una entidad del tipo B solo puede estar relacionada con una entidad del tipo A.

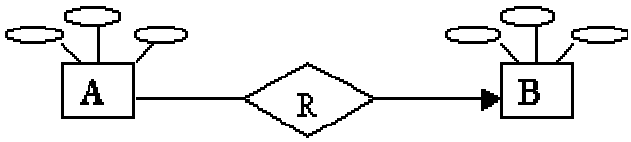
Su representación gráfica es la siguiente:



Nótese en este caso que el extremo punteado de la flecha de la relación de A y B, indica una entidad A conectada a muchas entidades B.

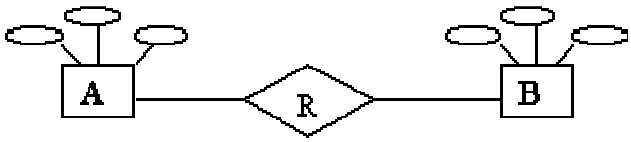
- ***Muchos a uno.***

Indica que una entidad del tipo B puede relacionarse con cualquier cantidad de entidades del tipo A, mientras que cada entidad del tipo A solo puede relacionarse con solo una entidad del tipo B.



- ***Muchas a muchas.***

Establece que cualquier cantidad de entidades del tipo A pueden estar relacionados con cualquier cantidad de entidades del tipo B.



A los tipos de relaciones antes descritos, también se le conoce como ***cardinalidad.***

La cardinalidad nos especifica los tipos de relaciones que existen entre las entidades en el modelo E-R y establecer con esto las validaciones necesarias para conseguir que los datos de la instancia (valor único en un momento dado de una base de datos) correspondan con la realidad.

Algunos ejemplos de cardinalidades de la vida común pueden ser:

***Uno a uno**.*

El noviazgo, el RFC de cada persona, El CURP personal, El acta de nacimiento, ya que solo existe un solo documento de este tipo para cada una de las diferentes personas.

***Uno a muchos**.*

Cliente – Cuenta en un banco, Padre-Hijos, Camión-Pasajeros, zoologico- animales, árbol – hojas.

Muchos a muchos.

Arquitecto – proyectos, fiesta – personas, estudiante – materias.

NOTA:

Cabe mencionar que la cardinalidad para cada conjunto de entidades depende del punto de vista que se le de al modelo en estudio, claro esta, sujetándose a la realidad.

Otra clase de limitantes lo constituye la ***dependencia de existencia.*** Refiriéndonos a las mismas entidades A y B, decimos que si la entidad A depende de la existencia de la entidad B, entonces A es dependiente de existencia por B, si eliminamos a B tendríamos que eliminar por consecuente la entidad A, en este caso B es la entidad *Dominante* y A es la entidad *subordinada.*

Llaves primarias.

Como ya se ha mencionado anteriormente, la distinción de una entidad entre otra se debe a sus atributos, lo cual lo hacen único. Una ***llave primaria*** es aquel atributo el cual consideramos clave para la identificación de los demás atributos que describen a la entidad. Por ejemplo, si consideramos la entidad ALUMNO del Instituto Tecnológico de La Paz, podríamos tener los siguientes atributos: Nombre, Semestre, Especialidad, Dirección, Teléfono, Número de control, de todos estos atributos el que podremos designar como llave primaria es el número de control, ya que es diferente para cada alumno y este nos identifica en la institución.

Claro que puede haber más de un atributo que pueda identificarse como llave primaria en este caso se selecciona la que consideremos más importante, los demás atributos son denominados ***llaves secundarias*.**

Una clave o llave primaria es indicada gráficamente en el modelo E-R con una línea debajo del nombre del atributo.

Diagrama Entidad-Relación

Denominado por sus siglas como: E-R; Este modelo representa a la realidad a través de un esquema gráfico empleando los terminología de ***entidades***, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas **atributos,** el enlace que rige la unión de las entidades esta representada por la **relación** del modelo.

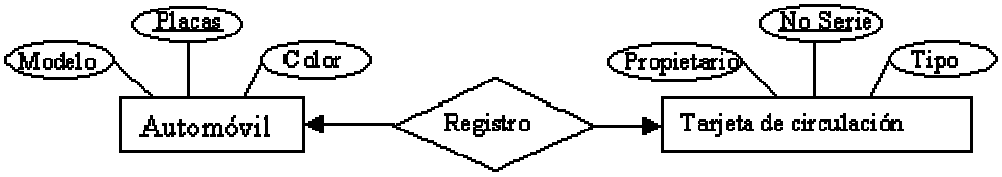
Recordemos que un rectángulo nos representa a las entidades; una elipse a los atributos de las entidades, y una etiqueta dentro de un rombo nos indica la relación que existe entre las entidades, destacando con líneas las uniones de estas y que la llave primaria de una entidad es aquel atributo que se encuentra subrayado.

A continuación mostraremos algunos ejemplos de modelos E-R, considerando las cardinalidades que existen entre ellos:

Relación Uno a Uno.

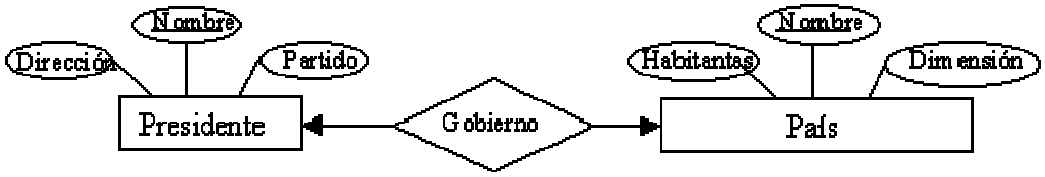
Problema:

Diseñar el modelo E-R, para la relación Registro de automóvil que consiste en obtener la tarjeta de circulación de un automóvil con los siguientes datos:- Automóvil- Modelo, Placas, Color - Tarjeta de circulación -Propietario, No_serie, Tipo.



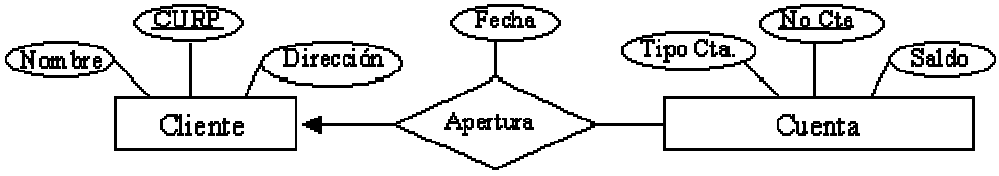
Indicamos con este ejemplo que existe una relación de pertenencia de uno a uno, ya que existe una tarjeta de circulación registrada por cada automóvil.

En este ejemplo, representamos que existe un solo presidente para cada país.



Relación muchos a muchos.

El siguiente ejemplo indica que un cliente puede tener muchas cuentas, pero que una cuenta puede llegar a pertenecer a un solo cliente (Decimos puede, ya que existen cuentas registradas a favor de más de una persona).

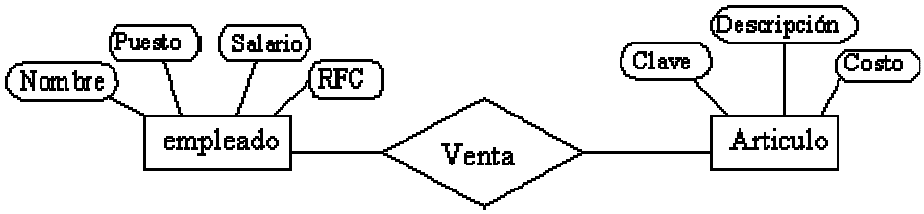


Reducción de diagramas E-R a tablas

Un diagrama E-R, puede ser representado también a través de una colección de tablas. Para cada una de las entidades y relaciones existe una tabla única a la que se le asigna como nombre el del conjunto de entidades y de las relaciones respectivamente, cada tabla tiene un número de columnas que son definidas por la cantidad de atributos y las cuales tienen el nombre del atributo.

La transformación de nuestro ejemplo Venta en la que intervienen las entidades de Vendedor con los atributos RFC, nombre, puesto, salario y Artículo con los atributos Clave, descripción, costo.

Cuyo diagrama E-R es el siguiente:



Entonces las tablas resultantes siguiendo la descripción anterior son:

Tabla Empleado

Nombre	Puesto	Salario	RFC
Teófilo	Vendedor	2000	TEAT701210XYZ
Cesar	Auxiliar ventas	1200	COV741120ABC

Tabla artículo

Clave	Descripción	Costo
A100	Abanico	460
C260	Colcha matrimonial	1200

Tabla Venta

RFC	Clave
TEAT701210XYZ	C260
COV741120ABC	A100

Nótese que en la tabla de relación - Venta -, contiene como atributos a las llaves primarias de las entidades que intervienen en dicha relación, en caso de que exista un atributo en las relaciones, este atributo es anexado como una fila más de la tabla;

Por ejemplo si anexamos el atributo fecha a la relación venta, la tabla que se originaria sería la siguiente:

RFC	Clave	Fecha
TEAT701210XYZ	C260	10/12/96
COV741120ABC	A100	11/12/96

Generalización y especialización

Generalización.

Es el resultado de la unión de 2 o más conjuntos de entidades (de bajo nivel) para producir un conjunto de entidades de más alto nivel. La generalización se usa para hacer resaltar los parecidos entre tipos de entidades de nivel más bajo y ocultar sus diferencias.

La generalización consiste en identificar todos aquellos atributos iguales de un conjunto de

entidades para formar una entidad(es) global(es) con dichos atributos semejantes, dicha entidad(es) global(es) quedara a un nivel más alto al de las entidades origen.

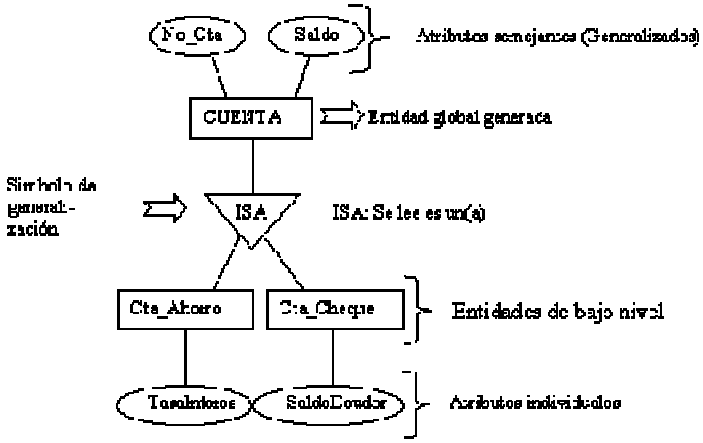
Ejemplo:

Tomando el ejemplo del libro de fundamentos de base de datos de Henry F. Korth.

Donde:

Se tiene las entidades Cta_Ahorro y Cta_Cheques, ambas tienen los atributos semejantes de No_Cta y Saldo, aunque además de estos dos atributos, Cta_Ahorro tiene el atributo Tasa_Interes y Cta_Cheques el atributo Saldo_Deudor. De todos estos atributos podemos juntar (generalizar) No_Cta y Saldo que son iguales en ambas entidades.

Entonces tenemos:



Podemos leer esta gráfica como: La entidad Cta_Ahorro hereda de la entidad CUENTA los atributos No_Cta y saldo, además del atributo de TasaInteres, de forma semejante Cta_cheque tiene los atributos de No_Cta, Saldo y SaldoDeudor.

Como podemos observar la Generalización trata de eliminar la redundancia (repetición) de atributos, al englobar los atributos semejantes. La entidad(es) de bajo nivel cuentan (heredan) todos los atributos correspondientes.

Especialización:

Es el resultado de tomar un subconjunto de entidades de alto nivel para formar un conjunto de entidades de más bajo nivel.

* En la generalización cada entidad de alto nivel debe ser también una entidad de bajo nivel. La especialización no tiene este limitante.

* se representa por medio de un triángulo denominado con la etiqueta "ISA", se distingue de la generalización por el grosor de las líneas que conectan al triángulo con las entidades.

* La especialización denota la diferencia entre los conjuntos de entidades de alto y bajo nivel.

Agregación.

La agregación surge de la limitación que existe en el modelado de E-R, al no permitir expresar las relaciones entre relaciones de un modelo E-R en el caso de que una relación X se quiera unir con una entidad cualquiera para formar otra relación.

La Generalización consiste en agrupar por medio de un rectángulo a la relación (representada por un rombo) junto con las entidades y atributos involucrados en ella, para formar un grupo que es considerado una entidad y ahora sí podemos relacionarla con otra entidad.

Para ejemplificar lo anterior consideremos el ejemplo del libro de fundamentos de Base de Datos de Henry F. Korth. En donde el problema consiste en que existen trabajando muchos empleados que trabajan en diferentes proyectos, pero dependiendo del trabajo que realiza en pueden llegar a utilizar un equipo o maquinaria; en este problema intervienen 3 entidades: Empleado, Proyecto y Maquinaria, el diagrama E-R correspondiente es:

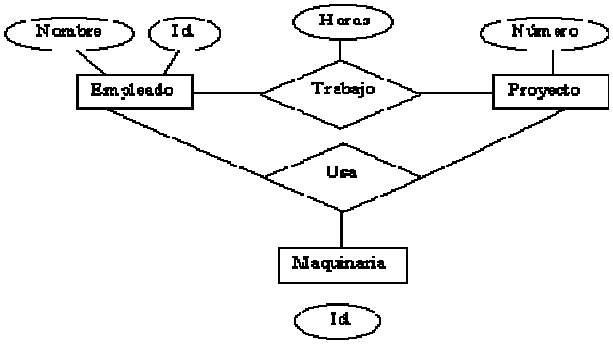


Diagrama E-R con relaciones redundantes

Como el modelo E-R no permite la unión entre dos o más relaciones, la relación trabajo es englobada como si fuera una entidad más de la relación usa, gráficamente queda como:

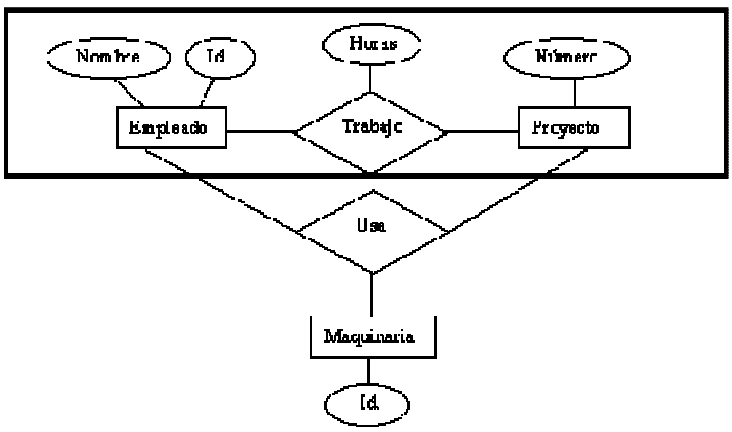


Diagrama E-R con agregación

Ahora podemos decir que la entidad trabajo se relaciona con la entidad maquinaria a través de la relación usar. Para indicarnos que un trabajo usa un determinado equipo o maquinaria según el tipo de trabajo que se trate.

MODELO RELACIONAL

La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las filas y las tablas son visibles en los datos. Este enfoque permite a los usuarios obtener información de la base de datos sin asistencia de sistemas profesionales de administración de información.

Las características más importantes de los modelos relacionales son:

a. Es importante saber que 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.

b. Todas las entradas de cualquier columna son de un solo tipo. Por ejemplo, una columna puede contener nombres de clientes, y en otra puede tener fechas de nacimiento. 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.

c. No existen 2 filas en la tabla que sean idénticas.

d. La información en las bases de datos son representados como datos explícitos, no existen apuntadores o ligas entre las tablas.

En el enfoque relacional es sustancialmente distinto de otros enfoques en términos de sus estructuras lógicas y del modo de las operaciones de entrada/salida. 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.

Así, una relación se compone de una colección de entidades(o registros) cuyos propietarios están descritos por cierto número de atributos predeterminados implantados como campos.

Estructura de las bases de datos relacionales

La arquitectura relacional se puede expresar en términos de tres niveles de abstracción: nivel interno, conceptual y de visión.

La arquitectura relacional consta de los siguientes componentes:

1. **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.

2. **Submodelo de datos:**

   Los esquemas externos de un sistema relacional se llaman submodelos 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 ("Area 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.

3. **Esquema de almacenamiento:**

   En el nivel interno, cada tabla base se implanta como un archivo almacenado. Para las recuperaciones sobre las claves principal o secundaria se pueden establecer uno o más índices para accesar un archivo almacenado.

4. **Sublenguaje 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 u solo comando del sublenguaje. Por tanto, el modo de operación de entrada/Salida en un sistema relacional se puede procesar en la forma: 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 sublenguaje de datos.

Lenguajes de consulta formales.

Los lenguajes de consultas:

Son los lenguajes en el que los usuarios solicitan información de la base de datos. Estos lenguajes son generalmente de más alto nivel que los lenguajes de programación. Los lenguajes de consulta pueden clasificarse como ***procedimentales*** y ***no procedimentales***;

En el lenguaje del tipo *procedimental* el usuario da las instrucciones al sistema para que realice una secuencia de operaciones en la base de datos para calcular el resultado deseado.

En el lenguaje *no procedimental,* el usuario describe la información deseada sin dar un procedimiento específico para obtener dicha información.

El álgebra relacional es un lenguaje de consulta formal procedimental, el álgebra relacional define operadores que funcionan sobre las tablas (de una manera similar a los operadores +,-,etc. del álgebra común) para llegar al resultado deseado. El álgebra relacional es difícil de utilizar, debido en parte a que es procedimental, esto es, al utilizar el álgebra relacional no sólo debemos ***saber*** lo que queremos, también ***cómo*** obtenerlo.

En el proceso de bases de datos comerciales el álgebra relacional se utiliza de manera poco frecuente. Aunque unos cuantos productos exitosos DBMS sí tienen opciones del álgebra relacional, éstas son poco utilizadas en vista de su complejidad.

El álgebra relacional toma dos o más tablas como entrada produce una nueva tabla como resultado de la serie de operaciones. Las operaciones fundamentales en el álgebra relacional son *seleccionar, proyectar, producto cartesiano, renombrar, unión y diferencia de conjuntos.* Además de las operaciones fundamentales existen otras operaciones como son: *intersección de conjuntos, producto natural, división y asignación.*

**** Operaciones fundamentales ****

Las operaciones seleccionar, proyectar y renombrar, son denominadas operaciones unitarias ya que operan sobre una tabla. Las otras operaciones operan sobre pares de relaciones y, por tanto se llaman operaciones binarias.

*** La operación seleccionar.**

Esta operación selecciona tuplas (filas) que satisfacen una instrucción(condición) dada de una tabla. Se representa por medio de paréntesis.

(nombre_tabla WHERE condición);

La oración de la instrucción después de la cláusula WHERE puede incluir condiciones de igualdad como =,=,<,>,>=,<=,además que se puede hacer una oración más compleja usando los conectores y (^) y o (v).

*** La operación Proyectar.**

Consiste en identificar las columnas (atributos en el modelo E-R) que nos interesa conocer. Se representa por medio de corchetes. Si este se omite indicara que se desea obtener todas las columnas de la tabla en cuestión.

(nombre_tabla WHERE condición) [Nombre_atributo];

*** La operación Producto cartesiano.**

Consiste en multiplicar todas las tuplas entre tablas, obteniendo como resultado una tabla que contiene todas las columnas de ambas tablas. Se especifica con la orden **TIMES.**

Nombre_tabla **TIMES** Nombre_tabla;

*** La operación Join.**

Consiste en obtener el producto (multiplicación) de todas las tuplas de una tabla con las de la otra, para posteriormente evaluar aquellas cuyo campo en común sea igual generando como resultado una nueva tabla que tiene como tuplas (renglones) que cumplen con la condición establecida. Se representa con la orden **JOIN**.

La orden Join es colocada entre las dos tablas a multiplicar después de que la primera especifica la operación de selección y proyección.

(Tabla)[atributo] **JOIN** (Tabla)[Atributo];

*** La operación Divide.**

Toma dos relaciones, una binaria y la otra unaria, construye una relación formada por todos los valores de un atributo de la relación binaria que concuerdan (en el otro atributo) con todos los valores de la relación unaria. Se representa con la orden **DIVIDEBY.**

NomTablaBin **DIVIDEBY** NomTablaUna

*** La operación Diferencia.**

Construye una relación formada por todas las tuplas (filas) de la primera relación que no aparezcan en la segunda de las dos relaciones especificadas. Se representa con la orden **MINUS.**

Nom_tablaA **MINUS** NomTablaB;

*** La operación Unión.**

Construye una relación formada por todas las tuplas de la primera relación y todas las tuplas de la segunda relación. El requisito es que ambas relaciones sean del mismo tipo.

Nom_TablaA **UNION** Nom_tablaB

*** La operación intersección.**

Construye una nueva tabla compuesta por todas las tuplas que están en la primera y segunda tabla**.**

Nom_TablaA **INTERSEC** Nom_tablaB

Ejemplos:

Para ejemplificar las notaciones anteriores consideremos el ejemplo ALUMNO - cursa - MATERIA, que tienen los siguientes atributos: NControl NControl Clave NombreA Clave NombreM Especialidad Calif Créditos Dirección

Representando en tablas a los atributos quedarían de la siguiente forma:

Tabla alumno:

NControl	NombreA	Especialidad	Dirección

Tabla cursa:

NControl	Clave	Calif

Tabla materia:

Clave	NombreM	Créditos

1.- Obtener el nombre de todos los alumnos que están inscritos en la Institución.

(Alumno) [NombreA];

2.- Obtener el nombre de los alumnos que cursan la materia Base de datos 1 cuya clave es SCB9333

(Alumno) JOIN (Cursa where Clave='SCB9333') [NombreA];

3.- Obtener los nombres de los alumnos de la especialidad de Ing. Sistemas que cursan la materia Base de datos 2.

((Alumno)[especialidad,NombreA,NControl]
JOIN (Cursa) where especialidad = 'ISC')[Clave,NombreA])
JOIN (Materia where NombreM='BD2')[NombreA];

En el álgebra relacional no solo debemos saber lo que queremos si no también como obtenerlo, al realizar las consultas debemos especificar el nombre de la tabla a utilizar en caso de que deseemos realizar una operación con un atributo que las otras tablas no tienen debemos "arrastrar" dicho atributo para poder utilizarlo, como lo es en el caso anterior, en donde requerimos el nombre del alumno que solamente lo tiene la tabla alumno, pro también deseamos que se cumpla la condición NombreM=BD2, como no podemos relacionar directamente a ambas tablas empleamos la tabla cursa de donde obtenemos la clave de las

materias y mantenemos el nombre del alumno (NombreA) finalmente con la orden JOIN se combinan las tablas por el campo común que tienen que es clave así que obtenemos una tabla con todas las materias que cursan los alumnos de ISC, de donde seleccionamos solo aquella que se llame BD2 con la orden Join obtenemos esta nueva tabla de donde por último proyectamos el atributo NombreA que hemos venido "arrastrando"**.**

Ejercicios propuestos:

Considere el modelo E-R del caso Médico - atiende - Paciente.

Realizar:

* La conversión a tablas del modelo E-R.
* Las siguientes consultas en álgebra relacional.

1.- Obtener el nombre de Todos los médicos**.**

2.- Obtener el nombre de todos los pacientes > de 18 años.

3.- Obtener todos los datos de todos los pacientes.

4.- Obtener los nombres de todos los pacientes que consultan con el médico con cédula profesional ABC001.

5.- Obtener los nombres de los médicos que atienden al paciente John Smith.

6.- Suponiendo que el hospital de la Ciudad de la Paz tiene una tabla de pacientes similar a la del hospital de San José, obtener el nombre y la afiliación de estos pacientes**.**

7.- Obtener las combinaciones de pacientes y médicos excepto la de aquellos médicos cuya especialidad sea Oftalmología.

Recuerde que tenemos que indicar las tablas a utilizar entre paréntesis y los atributos a proyectar entre corchetes, después podemos utilizar las ordenes Times, Join, Divide, Minus, Union, Intersec, según sea el caso a resolver; si requiere manipular atributos que no tengan las otras tablas "arrástrelos" proyectando siempre entre cada operación dicho atributo. De cada operación o combinación que realice entre las tablas se genera una tabla nueva que cumple con las condiciones que establece.

Lenguajes de consultas comerciales

Un lenguaje de consulta comercial proporciona una interfaz más amigable al usuario. Un ejemplo de este tipo de lenguaje es el SQL, (Structured Query Languaje, Lenguaje de Consulta Estructurado).

Las partes más importantes del SQL son:

DDL: Lenguaje de definición de datos (que nos permite crear las estructuras)

DML: Lenguaje de manipulación de datos (que nos permite tener acceso a las estructuras para suprimir, modificar e insertar)

En este apartado estudiaremos la forma básica para realizar consultas con SQL, en el apartado 3.4: Modificación de la base de datos, estudiaremos lo que concierne a la modificación de las tablas.

La estructura básica de una expresión en SQL contiene 3 partes, Select, From y Where.

La cláusula ***Select*** se usa para listar los atributos que se desean en el resultado de una consulta.

From, Lista las relaciones que se van a examinar en la evaluación de la expresión.

Where, es la definición de las condiciones a las que puede estar sujeta una consulta.

La consulta típica de SQL tiene la siguiente forma:

Select A1,A2,A3...An

From r1,r2,r3...rm

Where Condición(es)

Donde:

A1,A2,A3...An: Representan a cada atributo(s) o campos de las tablas de la base de datos relacional.
R1,r2,r3...rm: Representan a la(s) tabla(s) involucradas en la consulta.
Condición: Es el enunciado que rige el resultado de la consulta.

Si se omite la cláusula Where, la condición es considerada como verdadera, la lista de atributos (A1,A2..An) puede sustituirse por un asterisco (*), para seleccionar todos los atributos de todas las tablas que aparecen en la cláusula From.

Funcionamiento del SQL.

El SQL forma el producto cartesiano de las tablas involucradas en la cláusula From, cumpliendo con la condición establecida en la orden Where y después proyecta el resultado con la orden select.

Para nuestros ejemplos consideremos una tabla llamada CURSO, que contiene los siguientes campos:

Nombre del campo	Descripción
NumC	Número del curso, único para identificar cada curso
NombreC	Nombre del curso, también es único
DescC	Descripción del curso
Creditos	Créditos, número de estos que gana al estudiante al cursarlo
Costo	Costo del curso.
Depto	Departamento académico que ofrece el curso.

Datos contenidos en la tabla CURSO

NumC	NombreC	DescC	Creditos	Costo	Depto
A01	Liderazgo	Para público General	10	100.00	Admón.
S01	Introducción a la inteligencia artificial	Para ISC y LI	10	90.00	Sistemas.
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
B01	Situación actual y perspectivas de la alimentación y la nutrición	Para IB	8	80.00	Bioquímica
E01	Historia presente y futuro de la energía solar	IE e II	10	100.00	Electromecánica.
S02	Tecnología OLAP	Para ISC y LI	8	100.00	Sistemas
C02	Tecnología del concreto y de las Estructuras	Para IC	10	100.00	Ciencias
B02	Metabolismo de lípidos en el camarón	Para IB	10	0.00	Bioquímica
E02	Los sistemas eléctricos de potencia	Para IE	10	100.00	Electromecánica
S03	Estructura de datos	Para ISC y LI	8	0.00	Sistemas
A01	Diseño bioclimático	Para Arquitectura	10	0.00	Arquitectura
C03	Matemáticas discretas	General	8	0.00	Ciencias
S04	Circuitos digitales	Para ISC	10	0.00	Sistemas
S05	Arquitectura de Computadoras	Para ISC	10	50.00	Sistemas
I01	Base de Datos Relacionales	Para ISC y LI	10	150.00	Informática

Ejemplos de consultas:

OBTENCIÓN DE UNA TABLA ENTERA

- Obtener toda la información disponible sobre un curso donde Costo sea 0.

```
SELECT *
FROM CURSO
WHERE Costo=0.00
```

Resultado de la consulta anterior.

NumC	NombreC	DescC	Creditos	Costo	Depto
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
B02	Metabolismo de lípidos en el camarón	Para IB	10	0.00	Bioquímica
S03	Estructura de datos	Para ISC y LI	8	0.00	Sistemas
A01	Diseño bioclimático	Para Arquitectura	10	0.00	Arquitectura
C03	Matemáticas discretas	General	8	0.00	Ciencias

Colocamos un * debido a que no nos limitan la información de la tabla, es decir nos piden que mostremos todos los datos atributo de la tabla CURSO.

Como la única condición en la sentencia WHERE es que la tarifa del curso sea igual a 0, esta consulta regresa todas las tuplas donde se encuentre que Costo = 0.00.

Debido a que Costo es un campo numérico, la condición solo puede comparar con campos del mismo tipo. Para representar valores negativos se antepone a la izquierda el signo (-), en este ejemplo se considera solo el signo (=) para establecer la condición, sin embargo otros operadores que se pueden utilizar son:

Menor que <

Mayor que >

Menor o igual que <=

Mayor o igual que >=

Diferente <>

Además de los operadores booleanos AND, NOT, OR.

Cabe señalar que en la sentencia Where cuando se requiere establecer condiciones con cadenas, estas son delimitadas por apóstrofos (‘’). Las expresiones de cadenas son comparadas carácter por carácter, dos cadenas son iguales solo si coinciden todos los caracteres de las mismas.

Ejemplos de consultas con cadenas:

- Obtener toda la información sobre cualquier curso que ofrezca el departamento de Ciencias.

```
SELECT *
FROM CURSO
WHERE Depto = 'Ciencias';
```

Resultado de la consulta.

NumC	NombreC	DescC	Creditos	Costo	Depto
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
C02	Tecnología del concreto y de las Estructuras	Para IC	10	100.00	Ciencias
S04	Circuitos digitales	Para ISC	10	0.00	Sistemas

VISUALIZACIÓN DE COLUMNAS ESPECIFICADAS.

En los ejemplos anteriores obteníamos toda la tabla completa, ahora veremos como mostrar solo algunos atributos específicos de una tabla.

- Obtener los valores NumC,NombreC y Depto, en este orden de toda la tabla curso.

```
SELECT NumC, NombreC, Depto

FROM CURSO;
```

Resultado de la consulta:

NumC	NombreC	Depto
A01	Liderazgo	Admón.
S01	Introducción a la inteligencia artificial	Sistemas.
C01	Construcción de torres	Ciencias
B01	Situación actual y perspectivas de la alimentación y la nutrición	Bioquímica
E01	Historia presente y futuro de la energía solar	Electromecánica.
S02	Tecnología OLAP	Sistemas
C02	Tecnología del concreto y de las Estructuras	Ciencias
B02	Metabolismo de lípidos en el camarón	Bioquímica

E02	Los sistemas eléctricos de potencia	Electromecánica
S03	Estructura de datos	Sistemas
A01	Diseño bioclimático	Arquitectura
C03	Matemáticas discretas	Ciencias
S04	Circuitos digitales	Sistemas
S05	Arquitectura de Computadoras	Sistemas
I01	Base de Datos Relacionales	Informática

Observamos que en este caso no se tiene la sentencia Where, no existe condición, por lo tanto, todas las filas de la tabla CURSO se recuperan, pero solo se visualizaran las tres columnas especificadas.

Así mismo, empleamos la (,) para separar los campos que deseamos visualizar.

VISUALIZACIÓN DE UN SUBCONJUNTO DE FILAS Y COLUMNAS

- Seleccionar los valores NumC, Depto y Costo para todos los cursos que tengan un Costo inferior a $100

```
SELECT NumC, Depto, Costo
FROM CURSO
WHERE Costo < 100.00
```

Como resultado de esta consulta se obtendrán todas aquellas tuplas que tengan un costo en CTARIFA menor que 100, y se visualizaran solo los campos de NumC, Depto,Costo.

Podemos observar que este ejemplo cubre el formato general de una consulta SQL.

La palabra clave **DISTINCT**

DISTINCT, es una palabra reservada que elimina las filas que duplicadas en el resultado de una consulta.

- Visualizar todos los departamentos académicos que ofrezcan cursos, rechazando los valores duplicados.

```
SELECT DISTINCT Depto
FROM CURSO;
```

Resultado de la consulta

Depto

Administración
Sistemas
Ciencias
Bioquímica
electromecánica
Arquitectura
Informática

La palabra DISTINCT va estrictamente después de la palabra SELECT.

De no haberse utilizado la palabra DISTINCT, el resultado hubiera mostrado todas las tuplas del atributo Depto que se encontraran, es decir, se hubiera visualizado la columna de Depto completamente.

EMPLEO DE LOS CONECTORES BOOLEANOS (AND, OR, NOT)

Para emplear las condiciones múltiples dentro de la sentencia WHERE, utilizamos los conectores lógicos.

El conector **AND.**

Este conector pide al sistema que seleccione una sola columna únicamente si ambas condiciones se cumplen.

- Obtener toda la información sobre todos los cursos que ofrece el departamento Sistemas que tengan una tarifa igual a 0.

```
SELECT *

FROM CURSO

WHERE Depto='Sistemas' AND Costo=0.00;
```

El resultado de esta consulta sería todas aquellas tuplas que cumplan exactamente con las dos condiciones establecidas.

El conector **OR.**

Este conector al igual que el AND permite conectar condiciones múltiples en la sentencia WHERE, a diferencia del conector AND, el OR permite la selección de filas que cumplan con una sola de las condiciones establecidas a través de este conector.

- Obtener toda la información existente sobre cualquier curso ofrecido por los departamentos Arquitectura o Bioquímica.

```
SELECT *
FROM CURSO
WHERE Depto = 'Arquitectura' OR Depto= 'Bioquímica';
```

El resultado de esta consulta será la de visualizar todas aquellas tuplas donde se cumpla cualquiera de las 2 condiciones, es decir mostrara todas las tuplas que tengan en el atributo Depto=Arquitectura o Bioquímica.

El conector **NOT**

Este nos permite marcar aquellas tuplas que por alguna razón no deseamos visualizar.

- Obtener el nombre del curso y del departamento de todos los cursos que no sean ofrecidos por el departamento Sistemas.

SELECT NombreC, Depto
FROM CURSO
WHERE **NOT (**Depto='Sistemas'**);**

JERARQUIA DE OPERADORES BOOLEANOS.

En orden descendente (de mayor a menor prioridad)

- NOT
- AND
- OR

Existen dos formas para realizar consultas: Join de Querys y Subquerys.

Cuando en la sentencia **From** colocamos los nombres de las tablas separados por comas se dice que efectuamos una consulta de la forma **Join de Querys**, en este caso se requiere anteponer el nombre de la tabla y un punto al nombre del atributo. En el Join de Querys el resultado que se produce con las tablas que intervienen en la consulta es la concatenación de las tablas, en donde los valores de una columna de la primera tabla coinciden con los valores de una segunda tabla, la tabla de resultado tiene una fila por cada valor coincidente que resulte de las dos tablas originales.

Para ejemplificar esto, consideremos 2 tablas: Tabla1 y Tabla2, entonces:

C1	**C2**	**C3**		**CA**	**CB**
A	AAA	10		35	R
B	BBB	45		10	S
C	CCC	55		65	T
D	DDD	20		20	U
E	EEE	20		90	V
F	FFF	90		90	W
G	GGG	15		75	X
H	HHH	90		90	Y

				35	Z

Resultado de la operación Join:

C1	C2	C3	CA	CB
A	AAA	10	10	S
D	DDD	20	20	U
E	EEE	20	20	U
F	FFF	90	90	V
F	FFF	90	90	W
F	FFF	90	90	Y
H	HHH	90	90	V
H	HHH	90	90	W
H	HHH	90	90	Y

Como podemos observar, la comparación se efectuó por las columnas C3 y CA, que son donde se encontraron valores iguales, el resultado muestra una tupla por cada coincidencia encontrada.

Cuando las consultas se anidan se conoce como **Subquerys** o subconsultas. Este tipo de consulta obtiene resultados parciales reduciendo el espacio requerido para realizar una consulta.

Nota: Todas las consultas que se resuelven con subquerys pueden resolverse con Join de Querys, pero no todas las consultas hechas con Join de Querys pueden resolverse utilizando Subquerys.

Para ejemplificar lo anterior consideremos el ejemplo

ALUMNO - cursa - MATERIA, que tienen los siguientes atributos: NControl NControl Clave NombreA Clave NombreM Especialidad Calif Creditos Dirección

Representando en tablas a los atributos quedarían de la siguiente forma:

Tabla alumno:

NControl	NombreA	Especialidad	Dirección

Tabla cursa:

NControl	Clave	Calif

Tabla materia:

Clave	NombreM	Creditos

- Obtener el nombre de la materia que cursa el alumno con número de control 97310211 con créditos igual a ocho.

```
SELECT NombreA
FROM Materia
WHERE creditos='8' and clave in(SELECT clave
                                FROM cursa
                                WHERE   NControl='97310211';
```

- Obtener el número de control del alumno que tenga alguna calificación igual a 100

```
SELECT DISTINC(NControl)
FROM Cursa
WHERE Calif='100';
```

- Obtener el nombre de las materias que cursa el alumno Salvador Chávez.

```
SELECT NombreM
FROM Materia
WHERE Clave in (SELECT DISTINC (Clave)
                FROM Cursa
                WHERE NControl in (SELECT NControl)
                                  FROM Alumno
                                  WHERE NombreA='Salvador
                                                Chávez'));
```

FUNCIONES AVANZADAS APLICABLES A CONSULTAS

Existen funciones que permiten la agilización de consultas similares a una hoja de cálculo, ya que trabajan en base a renglones y columnas.

COUNT ( ): Cuenta el número de tuplas en la columna establecida

MIN ( ): Localiza el valor mínimo de la columna establecida

MAX ( ): Localiza el valor máximo de la columna establecida.

AVG ( ): Obtiene el promedio de valores de la columna establecida

SUM ( ): Obtiene el valor total que implican los valores obtenidos en la columna establecida.

Ejemplos:

- Obtener el número de alumnos que existen en la carrera de Ingeniería en Sistemas Computacionales.

```
SELECT Count (*)
FROM Alumno
WHERE especialidad='ISC';
```

- Obtener la máximo calificación que ha obtenido J.M. Cadena.

```
SELECT Max(Calif)
FROM Cursa
WHERE NControl IN (SELECT NControl
                   FROM Alumno
                   WHERE NombreA= 'J.M. Cadena ');
```

- Obtener el promedio de calificaciones de Salvador Chávez.

```
SELECT Avg (Calif)
FROM Cursa
WHERE NCotrol IN (SELECT NControl
                  FROM Alumno
                  WHERE NombreA='Salvador Chávez');
```

- Obtener la suma total de las calificaciones obtenidas por Daniel Colín.

```
SELECT Sum (Calif)
FROM Cursa
WHERE NControl IN (SELECT NControl
                   FROM Alumno
                   WHERE NombreA='Daniel Colín');
```

Hasta aquí hemos visto el manejo sencillo de realizar consultas con SQL, hay que destacar que en la realización de consultas anidadas se tiene que poner cuidando a la prioridad de los operadores, teniendo cuidado también al momento de agrupar los paréntesis que involucran las condiciones con los operadores.

Peligros en el diseño de bases de datos relacionales.

Uno de los retos en el diseño de la base de datos es el de obtener una estructura estable y lógica tal que:

1. El sistema de base de datos no sufra de anomalías de almacenamiento.
2. El modelo lógico pueda modificarse fácilmente para admitir nuevos requerimientos.

Una base de datos implantada sobre un modelo bien diseñado tiene mayor esperanza de vida aun en un ambiente dinámico, que una base de datos con un diseño pobre. En promedio, una base de datos experimenta una reorganización general cada seis años, dependiendo de lo dinámico de los requerimientos de los usuarios. Una base de datos bien diseñada tendrá un buen desempeño aunque aumente su tamaño, y será lo suficientemente flexible para incorporar nuevos requerimientos o características adicionales.

Existen diversos riesgos en el diseño de las bases de datos relacionales que afecten la funcionalidad de la misma, los riesgos generalmente son la redundancia de información y la inconsistencia de datos.

La normalización es el proceso de simplificar la relación entre los campos de un registro. Por medio de la normalización un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

- Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.

- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.

- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.

- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

En términos más sencillos la normalización trata de simplificar el diseño de una base de datos, esto a través de la búsqueda de la mejor estructuración que pueda utilizarse con las entidades involucradas en ella.

Pasos de la normalización:

1. Descomponer todos los grupos de datos en registros bidimensionales.
2. Eliminar todas las relaciones en la que los datos no dependan completamente de la llave primaria del registro**.**
3. Eliminar todas las relaciones que contengan dependencias transitivas.

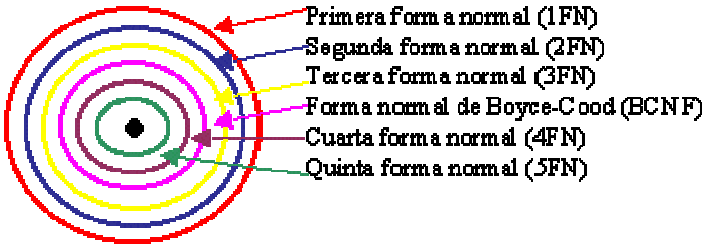
La teoría de normalización tiene como fundamento el concepto de formas normales; se dice que una relación está en una determinada forma normal si satisface un conjunto de restricciones.

Primera y segunda forma normal.

Formas normales.

Son las técnicas para prevenir las anomalías en las tablas. Dependiendo de su estructura, una tabla puede estar en primera forma normal, segunda forma normal o en cualquier otra.

Relación entre las formas normales:



Primera forma normal.

Definición formal:

Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valores atómicos.

Abreviada como 1FN, se considera que una relación se encuentra en la primera forma normal cuando cumple lo siguiente:

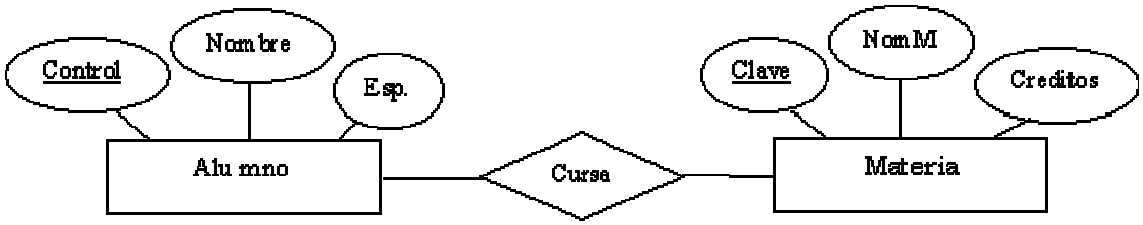
1. Las celdas de las tablas poseen valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
2. Todos los ingresos en cualquier columna(atributo) deben ser del mismo tipo.

3. Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.

4. Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no es importante.

Por lo general la mayoría de las relaciones cumplen con estas características, así que podemos decir que la mayoría de las relaciones se encuentran en la primera forma normal.

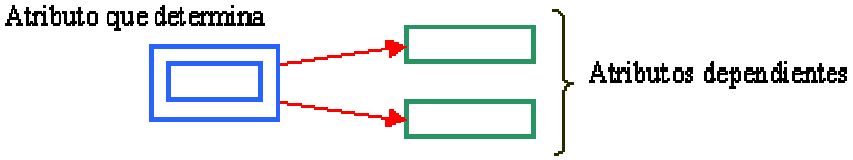
Para ejemplificar como se representan gráficamente las relaciones en primera forma normal consideremos la relación alumno cursa materia cuyo diagrama E-R es el siguiente:



Como esta relación maneja valores atómicos, es decir un solo valor por cada uno de los campos que conforman a los atributos de las entidades, ya se encuentra en primera forma normal, gráficamente así representamos a las relaciones en 1FN.

Segunda forma normal.

Para definir formalmente la segunda forma normal requerimos saber que es una **dependencia funcional:** Consiste en edificar que atributos dependen de otro(s) atributo(s).

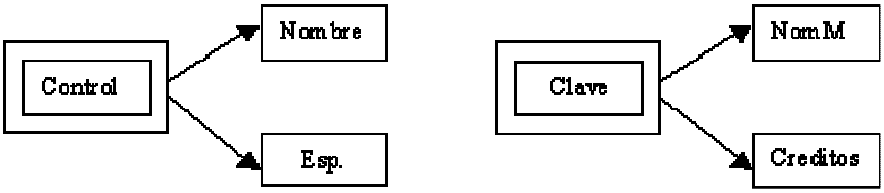


Definición formal:

Una relación R está en 2FN si y solo si está en 1FN y los atributos no primos dependen funcionalmente de la llave primaria.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves) dependen por completo de la clave . De acuerdo con está definición, cada tabla que tiene un atributo único como clave, esta en segunda forma normal.

La segunda forma normal se representa por dependencias funcionales como:



Nótese que las llaves primarias están representadas con doble cuadro, las flechas nos indican que de estos atributos se puede referenciar a los otros atributos que dependen funcionalmente de la llave primaria.

Tercera forma normal y la forma normal de Boyce Codd.

Para definir formalmente la 3FN necesitamos definir **dependencia transitiva:** En una afinidad (tabla bidimensional) que tiene por lo menos 3 atributos (A,B,C) en donde A determina a B**,** B determina a C pero no determina a A.

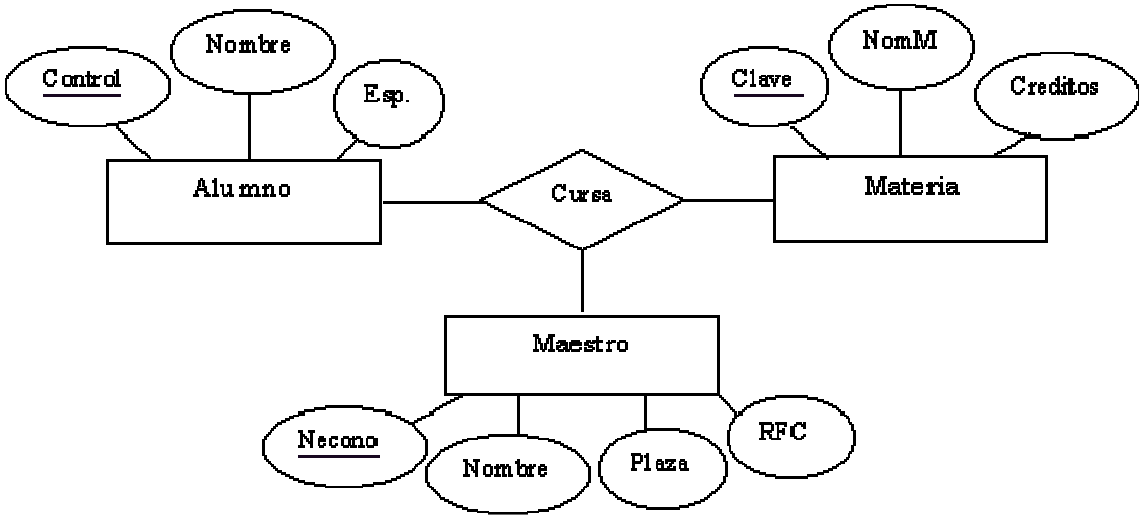
Tercera forma normal.

Definición formal:

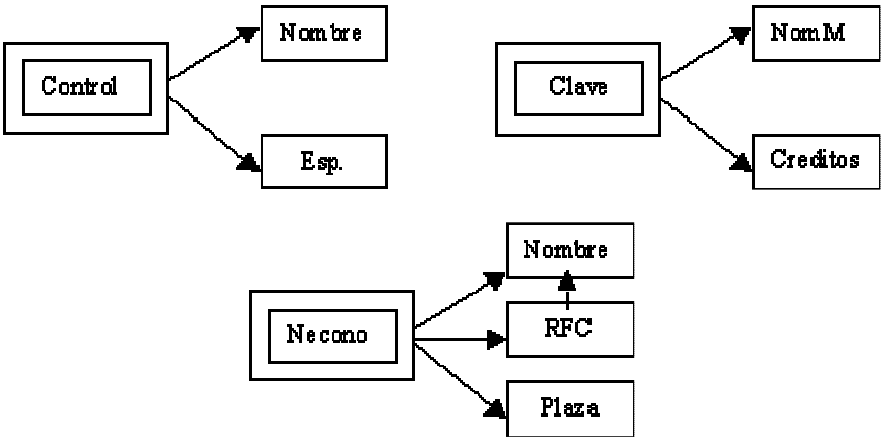
Una relación R está en 3FN si y solo si esta en 2FN y todos sus atributos no primos dependen no transitivamente de la llave primaria.

Consiste en eliminar la dependencia transitiva que queda en una segunda forma normal, en pocas palabras una relación esta en tercera forma normal si está en segunda forma normal y no existen dependencias transitivas entre los atributos, nos referimos a dependencias transitivas cuando existe más de una forma de llegar a referencias a un atributo de una relación.

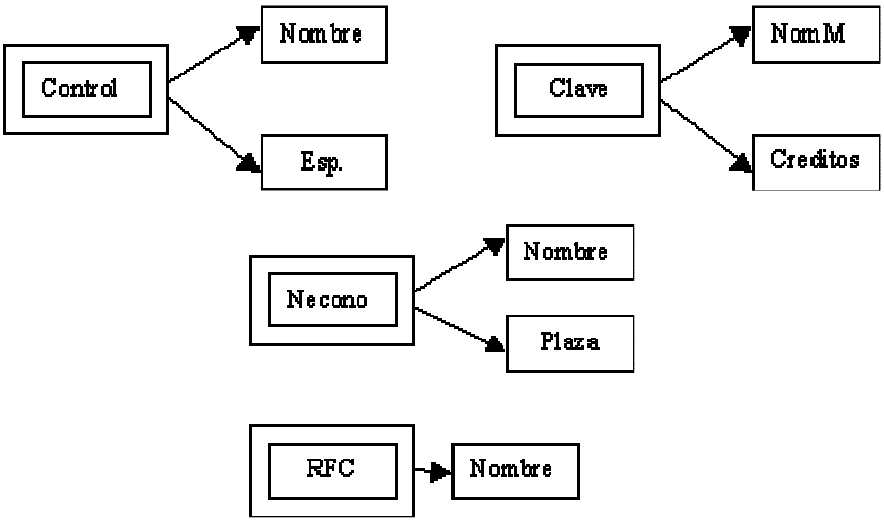
Por ejemplo, consideremos el siguiente caso:



Tenemos la relación alumno-cursa-materia manejada anteriormente, pero ahora consideramos al elemento maestro, gráficamente lo podemos representar de la siguiente manera:



Podemos darnos cuenta que se encuentra graficado en segunda forma normal, es decir que todos los atributos llave están indicados en doble cuadro indicando los atributos que dependen de dichas llaves, sin embargo en la llave Necono tiene como dependientes a 3 atributos en el cual el nombre puede ser referenciado por dos atributos: Necono y RFC (Existe dependencia transitiva), podemos solucionar esto aplicando la tercera forma normal que consiste en eliminar estas dependencias separando los atributos, entonces tenemos:



Forma normal de Boyce Codd.

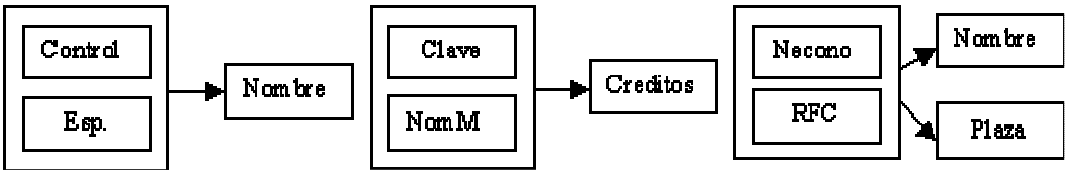
Determinante: Uno o más atributos que, de manera funcional, determinan otro atributo o atributos. En la dependencia funcional (A,B)-->C, (A,B) son los determinantes.

Definición formal:
Una relación R esta en FNBC si y solo si cada determinante es una llave candidato.

Denominada por sus siglas en ingles como BCNF; Una tabla se considera en esta forma si y sólo sí cada determinante o atributo es una llave candidato.

Continuando con el ejemplo anterior, si consideramos que en la entidad alumno sus atributos control y nombre nos puede hacer referencia al atributos esp., entonces decimos que dichos atributos pueden ser llaves candidato.

Gráficamente podemos representar la forma normal de Boyce Codd de la siguiente forma:



Obsérvese que a diferencia de la tercera forma normal, agrupamos todas las llaves candidato para formar una global (representadas en el recuadro) las cuales hacen referencia a los atributos que no son llaves candidato.

Cuarta y quinta forma normal

Cuarta forma normal.

Definición formal:

Un esquema de relaciones R está en 4FN con respecto a un conjunto D de dependencias funcionales y de valores múltiples sí, para todas las dependencias de valores múltiples en D de la forma X->->Y, donde X<=R y Y<=R, se cumple por lo menos una de estas condiciones:

* X->->Y es una dependencia de valores múltiples trivial.
* X es una superllave del esquema R.

Para entender mejor aún esto consideremos una afinidad (tabla) llamada estudiante que contiene los siguientes atributos: Clave, Especialidad, Curso tal y como se demuestra en la siguiente figura:

Clave	Especialidad	Curso
S01	Sistemas	Natación
S01	Bioquímica	Danza
S01	Sistemas	Natación
B01	Bioquímica	Guitarra
C03	Civil	Natación

Suponemos que los estudiantes pueden inscribirse en varias especialidades y en diversos cursos. El estudiante con clave S01 tiene su especialidad en sistemas y Bioquímica y toma los cursos de Natación y danza, el estudiante B01 tiene la especialidad en Bioquímica y toma el curso de Guitarra, el estudiante con clave C03 tiene la especialidad de Civil y toma el curso de natación.

En esta tabla o relación no existe dependencia funcional porque los estudiantes pueden tener distintas especialidades, un valor único de clave puede poseer muchos valores de especialidades al igual que de valores de cursos. Por lo tanto existe **dependencia de valores múltiples.** Este tipo de dependencias produce redundancia de datos, como se puede apreciar en la tabla anterior, en donde la clave S01 tiene tres registros para mantener la serie de datos en forma independiente lo cual ocasiona que al realizarse una actualización se requiera de demasiadas operaciones para tal fin.

Existe una dependencia de valores múltiples cuando una afinidad tiene por lo menos tres atributos, dos de los cuales poseen valores múltiples y sus valores dependen solo del tercer atributo, en otras palabras en la afinidad R (A,B,C) existe una dependencia de valores múltiples si A determina valores múltiples de B, A determina valores múltiples de C, y B y C son independientes entre sí.

En la tabla anterior Clave determina valores múltiples de especialidad y clave determina valores múltiples de curso, pero especialidad y curso son independientes entre sí.

Las dependencias de valores múltiples se definen de la siguiente manera: Clave ->->Especialidad y Clave->->Curso; Esto se lee "Clave multidetrmina a Especialidad, y clave multidetermina a Curso"

Para eliminar la redundancia de los datos, se deben eliminar las dependencias de valores múltiples. Esto se logra construyendo dos tablas, donde cada una almacena datos para solamente uno de los atributos de valores múltiples.

Para nuestro ejemplo, las tablas correspondientes son:

Tabla Eespecialidad

Clave	Especialidad
S01	Sistemas
B01	Bioquímica
C03	Civil

Tabla ECurso

Clave	Curso
S01	Natación
S01	Danza
B01	Guitarra
C03	Natación

Quinta forma normal.

Definición formal:

Un esquema de relaciones R está en 5FN con respecto a un conjunto D de dependencias funcionales, de valores múltiples y de producto, si para todas las dependencias de productos en D se cumple por lo menos una de estas condiciones:

* (R1, R2, R3, ... Rn) es una dependencia de producto trivial.
* Toda Ri es una superllave de R.

La quinta forma normal se refiere a dependencias que son extrañas. Tiene que ver con tablas que pueden dividirse en subtablas, pero que no pueden reconstruirse.