4.1. Concepto de archivo y directorio.

Concepto de archivo

Archivo (o fichero) es un conjunto de información sobre un mismo tema, tratada como una unidad de almacenamiento en memoria secundaria y organizada de forma estructurada para la búsqueda de un dato individual. Un archivo está compuesto de registros homogéneos que contienen información sobre el tema.

Al principio del disco se escribe la TOC (Tabla de Contenidos), que tiene el nombre de cada archivo y su dirección.

La vida del archivo comienza cuando se crea y acaba cuando se borra.

Un **registro** es una estructura dentro del archivo que contiene la información correspondiente a un elemento individual. Un registro se puede dividir en campos.

Un **campo** es un dato que representa una información unitaria o independiente.

Operaciones sobre archivos

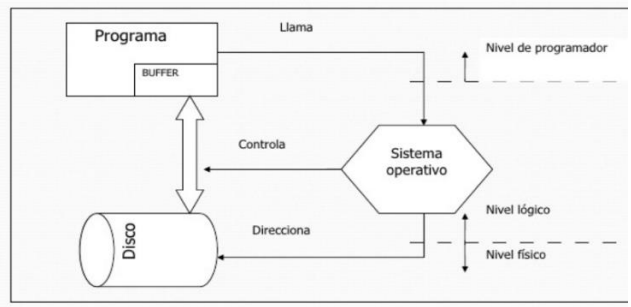
El sistema operativo permite que el usuario pueda aludir al archivo mediante un nombre, independientemente de la forma en que se almacene en el dispositivo (por ejemplo, un disco).

Todo archivo tiene asociados unos atributos, además del nombre, como: tamaño, fecha de creación y modificación, propietario, permisos de acceso, etc. (dependerá del SO).

El SO proporciona órdenes o servicios para operar sobre los archivos como:

- Crear / copiar / borrar / renombrar un archivo
- Establecer / obtener atributos de un archivo (*fecha de creación, quien lo creó*)
- Abrir / cerrar un archivo para el procesamiento de su contenido
- Leer / escribir un registro de un determinado archivo

Gestión de archivos



Gestión del acceso a archivos realizada por el sistema operativo

- El sistema operativo transporta, cada vez que se accede al dispositivo, una cantidad fija de información que depende de las características físicas de este: bloque o registro físico, de longitud distinta al tamaño del registro.
- El sistema operativo transforma la dirección lógica usada en los programas de usuario en la dirección física con la que se direcciona en el dispositivo.
- Un archivo es una estructura de datos externa al programa que lo usa; en las operaciones de lectura / escritura se transfiere información a / desde un buffer en memoria principal asociado a las operaciones de entrada / salida sobre el archivo.

Clasificación de archivos según el tipo de registros

- **Longitud fija**: todos los registros tienen la misma longitud (*los mismos campos y la misma longitud, el tamaño de los registros es el mismo*)
- **Longitud variable:** el sistema reserva una palabra al comienzo de cada registro para anotar su longitud
    - Con delimitador: un determinado carácter llamado delimitador marca el fin de un registro (suelen usarse como delimitadores el salto de línea, nulo, etc.)
    - Con cabecera: cada registro contiene un campo inicial que almacena el número de bytes del registro
- **Longitud indefinida:** el sistema operativo no realiza ninguna gestión sobre la longitud de los registros ya que el archivo no tiene realmente ninguna estructura interna.
    - En cada operación de lectura / escritura se transfiere una determinada subcadena del archivo y será el programa de usuario quien indique al SO el principio y el final de ese registro.

Una posibilidad añadida: disponer de un **campo clave** (o llave) que permita localizar rápidamente un registro.

Concepto de directorio

Directorio es un archivo especial que permite agrupar archivos según las necesidades de los usuarios. En cada dispositivo existe una estructura jerárquica donde se localizan todos los archivos de los distintos usuarios que lo usan.

En la visión que el usuario tiene de la estructura jerárquica de archivos se utilizan los siguientes conceptos:

- Directorio actual o de trabajo
- Directorio inicial o *home*
- Rutas (*pathname*) absolutas y relativas
- Lista de búsqueda
- Enlace duro, enlace simbólico

El sistema operativo proporciona operaciones para manejar los conceptos anteriores.

4.2 Organización de archivos

Organización secuencial

Los registros están almacenados físicamente de forma contigua, es decir, uno a continuación del anterior, siguiendo la secuencia lógica del archivo.

Para leer el cuarto registro, por ejemplo, tengo que leer el primero, el segundo y el tercero, aunque no me interesen. Se escriben en ese orden y se tienen que leer en orden.

Organización secuencial:

- *Física → dispositivo físicamente secuencial: casette*
- *Lógica*

La organización secuencial es adecuada:

- Cuando se quiere leer los registros en la misma secuencia en la que están almacenados
- Cuando se necesita leer la mayoría de los registros del archivo

Ventajas:

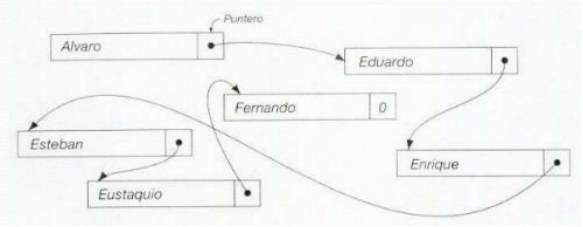
- Buen aprovechamiento del espacio
- Sencilla de utilizar
- Se puede utilizar con dispositivos secuenciales que son de bajo precio

Operaciones sobre archivos:

- **Añadir**: solo es posible escribir al final del archivo, después del último registro escrito
- **Consulta o recuperación**: se realiza en orden secuencial, es decir, el orden en el que se hubieran escrito determina el orden en el que se leen los registros. Para leer el registro que ocupa la posición n en el archivo es necesario leer previamente los n-1 registros que hay antes que él
- **Inserción, modificación y eliminación**: no es fácil realizar estas operaciones sobre un archivo secuencial
    - La modificación de un registro solo es posible si no se aumenta su longitud
    - No es posible eliminar un registro del archivo, pero se puede realizar un borrado lógico, es decir, marcarlo de tal forma que al leer se identifique como no válido
    - En otros casos es necesario crear un archivo nuevo con las actualizaciones que se quieran realizar

<u>Organización secuencial encadenada</u>

Junto a cada registro se almacena un puntero con la dirección física del registro siguiente, dando lugar a una cadena de registros. El último registro de la cadena contiene una marca especial en el lugar del puntero indicando que ya no hay más registros en el archivo.



Principal ventaja: facilidad de inserción y borrado de registros

Principal inconveniente: limita las consultas de forma secuencial

Operaciones sobre archivos:

- **Consulta o recuperación**. La consulta es secuencial, al igual que en un archivo con organización secuencial pura. Cada vez que se lee un registro se lee la posición del siguiente, lo que permite seguir la secuencia lógica del archivo

- **Inserción**. Hay que seguir los siguientes pasos:
    1. Localizar la posición donde se desea insertar; es decir, entre qué dos registros debe estar el nuevo registro del archivo
    2. Escribir el registro en una zona libre de memoria
    3. Asignar al nuevo registro como puntero la dirección física del registro siguiente
    4. Modificar el registro anterior para actualizar el valor de su puntero de forma que contenga la dirección del nuevo registro
    - Ejemplo de inserción

Dir. Física	LLave	Puntero
1	Alba	4
2	Elefante	3
3	León	8
4	Coral	2
5	Mosca	6
6	Ñu	7
7	Pantera	0
8	Morsa	5
9		
10		

Dir. Física	LLave	Puntero
1	Alba	4
2	Elefante	3
3	León	8
4	Coral	2
5	Mosca	6
6	Ñu	7
7	Pantera	0
8	Morsa	5
9	Gato	3
10		

Dir. Física	LLave	Puntero
1	Alba	4
2	Elefante	9
3	León	8
4	Coral	2
5	Mosca	6
6	Ñu	7
7	Pantera	0
8	Morsa	5
9	Gato	3
10		

- **Borrado**. Para borrar un registro se asigna al puntero del registro anterior la dirección del registro siguiente al que se desea borrar
- **Modificación**. Si el cambio no implica un aumento de longitud del registro, éste puede reescribirse en el mismo espacio. En caso contrario, se debe insertar el registro y luego borrar la versión anterior al cambio.

Organización secuencial indexada

Está ordenado por algún criterio

Un archivo con organización secuencial indexada está formado por dos estructuras:

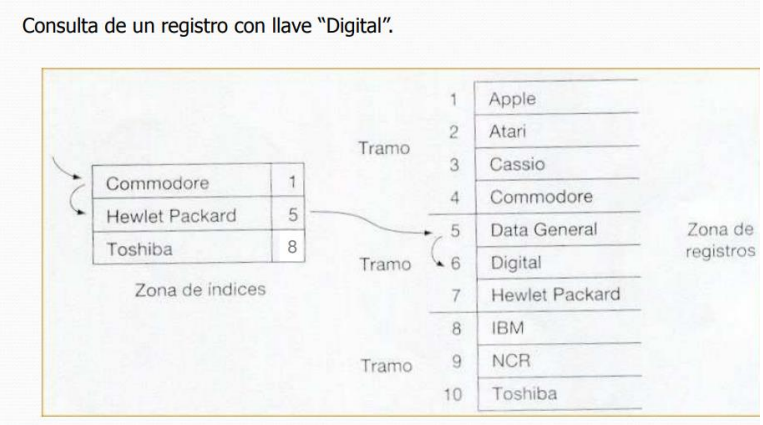
- **Zona de registros**: es una zona donde se direccionan los registros del archivo; está dividida en tramos (conjunto de registros consecutivos). Los registros están ordenados según el valor de una llave
- **Zona de índices**: es una zona en la que por cada tramo de la zona de registros hay un registro que contiene:
    - El mayor valor de la llave del tramo (valor de llave del último registro del tramo)
    - La dirección del primer registro del tramo

La gestión de la estructura la realiza el sistema operativo o un software especial, por lo que el usuario de esta estructura no necesita conocer la existencia de ambas zonas, pudiendo contemplar ambas como un todo.

Ejemplo

Operaciones sobre archivos:

- **Consulta**. Se realiza por llave (esto es, localizar un registro conocida su llave) sin necesidad de leer los registros que no se encuentran en su mismo tramo. El procedimiento a seguir para realizar la consulta por llave es:
    1. Leer secuencialmente las llaves en la zona de índices hasta encontrar una mayor o igual a la del registro buscado
    2. Obtener la dirección de comienzo del tramo donde está el registro
    3. Leer secuencialmente el tramo de zona de registros a partir de la dirección obtenida en la zona de índices hasta encontrar el registro buscado o uno con valor de la llave mayor que el buscado (en este último caso el registro no se encuentra en el archivo)
    - Ejemplo



- **Inserción**. Dado que ambas zonas son secuenciales, no es posible insertar un registro en archivos con esta organización
    - En algunos casos se permite la escritura de nuevos registros al final de la zona de registros. Estos registros, como es lógico, no podrán ser consultados por llave con el procedimiento descrito anteriormente
- **Borrado**. Al estar los registros escritos en secuencia no es posible borrar un registro
    - La única forma de eliminar la información contenida en un registro es marcándolo, lo que se conoce como borrado lógico
- **Modificación**. Las modificaciones son posibles tan solo si el registro no aumenta de longitud al ser modificado y no se altera el valor de la llave del mismo

Organización directa o aleatoria

Un archivo con organización directa o aleatoria (*random*), es un archivo escrito sobre un soporte de acceso directo para el cual existe una función de transformación que genera la dirección de cada registro en el archivo a partir de un campo que se usa como llave.

El nombre de aleatorio se debe a que normalmente no existe ninguna vinculación aparente entre el orden lógico de los registros y su orden físico.

La organización directa es útil para archivos donde los accesos deben realizarse por llave, accediéndose siempre a registros concretos.

Si la información se va a procesar en conjunto, con frecuencia puede ser más rentable una organización secuencial indexada.

Problema de los sinónimos

Un problema fundamental de esta organización es elegir adecuadamente la función de transformación o método de direccionamiento que se va a utilizar, ya que pueden darse las siguientes situaciones no deseadas:

- Que haya direcciones que no se corresponden con ninguna llave y, por tanto, habrá zonas de disco sin utilizar
- Que haya direcciones que se correspondan con más de una llave. En este caso, se dice que las llaves son sinónimas para esa transformación o que se produce una colisión

Hay dos formas de **resolver** el problema de los sinónimos, siempre a costa de complicar la estructura del archivo:

- Cuando se asocia a una llave una dirección ya ocupada por un registro distinto (esto es, por un sinónimo de esta llave), se busca en el archivo hasta encontrar una posición libre donde escribir el registro
- Se reserva una zona de desbordamiento donde se escribirán los registros que no se pueden escribir en la posición que les corresponde según la transformación. Esta zona se puede gestionar secuencialmente o encadenada a la zona principal de registros

Métodos de direccionamiento

1. **Direccionamiento directo**. Se utiliza como dirección la propia llave y solo es factible cuando la llave es numérica y su rango de valores no es mayor que el rango de direcciones en el archivo
    - Por ejemplo, el archivo de habitaciones de un hotel puede organizarse en forma aleatoria con direccionamiento directo haciendo coincidir la dirección con el número de habitación
- Inconveniente: en algunos casos pueden quedar lagunas de direcciones sin utilizar, en lugares conocidos de antemano. En este caso se pueden ocupar dichas direcciones desplazando las direcciones superiores
2. **Direccionamiento asociado**. Se puede utilizar cualquier tipo de llave. Si se utiliza este método debe construirse una tabla en la que se almacena para cada llave la dirección donde se encuentra el registro correspondiente. Dicha tabla se debe guardar mientras exista el archivo

3. **Direccionamiento calculado ("hashing")**. La dirección de cada registro se obtiene realizando una transformación sobre la llave. Se utiliza cuando:
    - La llave no es numérica, en cuyo caso se necesita una conversión previa para obtener un número a partir de ella
        - Por ejemplo, se usa el equivalente decimal al propio código binario del carácter (al carácter A le correspondería el 65)
    - La llave es numérica, pero toma valores en un rango inadecuado para usarse directamente como dirección

Operaciones sobre archivos

- **Consulta**. La consulta se realiza por llave. Para leer un registro debe aplicarse a la llave el algoritmo de transformación, que devuelve un número que es la dirección del registro que se quiere leer. Si el registro con la llave buscada no se encuentra allí, se procederá según se haya resuelto la gestión de sinónimos o colisiones
- **Borrado**. Siempre se realiza un borrado lógico, pudiéndose reutilizar el espacio del registro eliminado
- **Modificación e inserción**. Siempre se puede modificar o insertar un nuevo registro, realizando la transformación de la llave correspondiente

4.3. Bases de datos.

Problemática

En una aplicación convencional con archivos aparecen los siguientes problemas:

1. **Dificultad de mantenimiento**. Si hay archivos con información parcialmente duplicada, realizar las actualizaciones necesarias es un problema complejo y costoso. Normalmente, es necesario actualizar varios archivos con diferentes organizaciones. Si la actualización no se realiza correctamente se tendrá información incoherente
2. **Redundancia**. Se dice que hay redundancia si un dato se puede deducir a partir de otros datos (se dan los problemas del caso anterior)
3. **Rigidez de búsqueda**. El archivo se concibe para acceder a los datos de un modo determinado. Sin embargo, en la mayoría de los casos es necesario (o al menos deseable) combinar acceso secuencial y directo por varias claves
4. **Dependencia con los programas**. En un archivo no están reflejadas las relaciones existentes entre campos y registros. El programa que trabaja con el archivo es quien determina en cada caso dichas relaciones. En consecuencia, cualquier modificación de la estructura de un archivo obliga a modificar todos los programas que lo usen. Esto ocurre aun en el caso de que la alteración sea ajena al programa. Así, por ejemplo, si se aumenta la longitud de un campo habrá que modificar incluso los programas que no lo usan.
5. **Seguridad**. Uno de los mayores problemas de cualquier sistema de información es mantener la seguridad necesaria sobre los datos que contiene. Si se está trabajando con archivos, el control deberá realizarlo el propio programa. El aspecto de la seguridad es particularmente deficitario en los sistemas de archivos.

Concepto de base de datos

Las bases de datos surgen como alternativa a los sistemas de archivos, intentado eliminar o al menos reducir sus inconvenientes.

Una base de datos es un sistema formado por un conjunto de datos y un paquete software para la gestión de dicho conjunto de datos de tal modo que:

- Se controla el almacenamiento de datos redundantes
- Los datos resultan independientes de los programas que los usan
- Las relaciones entre los datos se almacenan junto a ellos
- Se puede acceder a los datos de diversas formas

La forma en la que se almacenan las relaciones entre datos y el utilizar como unidad de almacenamiento el campo además del registro, es el fundamento de la independencia respecto a los programas de aplicación.

Requisitos que deben cumplir las bases de datos

1. **Acceso múltiple**. Diversos usuarios pueden acceder a la base de datos, sin que se produzcan conflictos, ni versiones incoherentes
2. **Utilización múltiple**. Cada usuario podrá tener una imagen o visión particular de la estructura de la base de datos
3. **Flexibilidad**. Se podrán usar distintos métodos de acceso, con tiempos de respuesta razonablemente pequeños
4. **Seguridad**. Se controlará el acceso a los datos (a nivel de campo), impidiéndoselo a los usuarios no autorizados
5. **Protección contra fallos**. Deben existir mecanismos concretos de recuperación en caso de fallo de la computadora
6. **Independencia física**. Se puede cambiar el soporte físico de la base de datos sin que esto repercuta en la base de datos ni en los programas que la utilizan
7. **Independencia lógica**. Se pueden modificar los datos contenidos en la base, las relaciones inexistentes entre ellos o incluir nuevos datos sin afectar a los programas que la usan
8. **Redundancia controlada**. Los datos se almacenan una sola vez
9. **Interfaz de alto nivel**. Existe una forma sencilla y cómoda de utilizar la base al menos desde un lenguaje de programación de alto nivel
10. **Interrogación directa ("query")**. Existe una utilidad que permite el acceso a los datos de forma interactiva o conversacional

Estructura de una base de datos

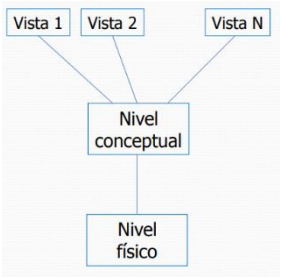
Se dice que uno o mas atributos de una entidad es un identificador o clave primaria si el valor de dichos atributos determina de forma unívoca cada uno de los elementos de dicha entidad, y no existe ningún subconjunto de él que permita identificar a la entidad de manera única.

En una base de datos se almacenan además de las entidades, las **relaciones** existentes entre ellas.

En la implementación de la base de datos, estas relaciones se almacenan con punteros que inserta automáticamente el software que la maneja y esto es “transparente al usuario”.

Niveles de abstracción de la información: vistas y esquemas

- **Nivel de vista.** Permite describir diferentes vistas o subesquemas, cada una de las cuales se corresponde con la parte de la base de datos que interesa a un determinado grupo de usuarios. Además, limita el acceso solo a la información de la vista.
- **Nivel conceptual.** Describe el esquema de la base de datos. En éste se especifica qué información se guarda en la base de datos, incluyendo todos los datos almacenados en ella y las relaciones entre ellos. Este nivel se utiliza en la administración de la base de datos
- **Nivel físico.** Describe cómo se almacenan los datos, con las estructuras de datos necesarias para ello



Modelos de datos

Modelo de datos: grupo de herramientas conceptuales que permite describir los datos, sus relaciones, su semántica y sus limitaciones. Ayuda a describir la estructura de una base de datos.

Clasificación de los modelos de datos:

- **Modelos lógicos basados en objetos**. Describen los datos a nivel conceptual y a nivel de vista, permiten una estructuración flexible y especificar limitaciones de los datos. Caso a tratar: modelo entidad-relación
- **Modelos lógicos basados en registros**. Describen los datos a nivel conceptual y a nivel de vista, permitiendo especificar la estructura lógica pero no las limitaciones de los datos. Casos a tratar: modelo jerárquico, modelo en red y modelo relacional
- **Modelos físicos de los datos**. Describen los datos en el nivel de implementación de los sistemas de base de datos

Modelo entidad-relación

Conceptos básicos

- **Entidad**. Objeto que tiene existencia propia, que puede distinguirse de otros y del cual se quiere almacenar información de ciertas características. *Ejemplo: Pepe con DNI 24324450 y que vive en Granada*
- **Conjunto de entidades.** Grupo de entidades del mismo tipo que representa la estructura genérica de una entidad de interés. Las entidades pueden pertenecer a más de un conjunto de entidades. *Ejemplo: Cliente*
- **Relación.** Asociación entre varias entidades. *Ejemplo: el cliente con DNI 24324450 compra un coche con matrícula 6670 BBC el 7/11/2005*
- **Conjunto de relaciones.** Grupo de relaciones del mismo tipo que representa la estructura genérica de las relaciones entre conjuntos de entidades. *Ejemplo: Compra*
- **Grado de una relación.** Numero de tipos de entidad que intervienen en un tipo de relación. *Ejemplo: en el caso de la relación en la que un cliente compra un coche, el grado de la relación es 2.*
- **Atributo.** Unidad básica de información sobre un tipo de entidad o un tipo de relación. *Ejemplos: DNI, nombre, dirección, fecha de compra.*

Tipos de correspondencias (cardinalidad)

Cardinalidad: expresa el numero de entidades con las que puede asociarse o corresponderse una determinada entidad mediante una relación.

En el caso de un conjunto binario de relaciones R entre los conjuntos de relaciones A y B, los tipos de correspondencias o cardinales pueden ser:

- **Uno a uno (1:1)**. A cada entidad de A le puede corresponder una única entidad de B y viceversa
- **Uno a muchos (1:N).** A cada entidad de A le puede corresponder más de una entidad de B, pero cada entidad de B solo puede asociarse con una única entidad de A
- **Muchos a uno (N:1).** A cada entidad de A le puede corresponder una única entidad de B, pero cada entidad de B puede asociarse con varias entidades de A
- **Muchos a muchos (N:M).** A cada entidad de A le puede corresponder más de una entidad de B y viceversa

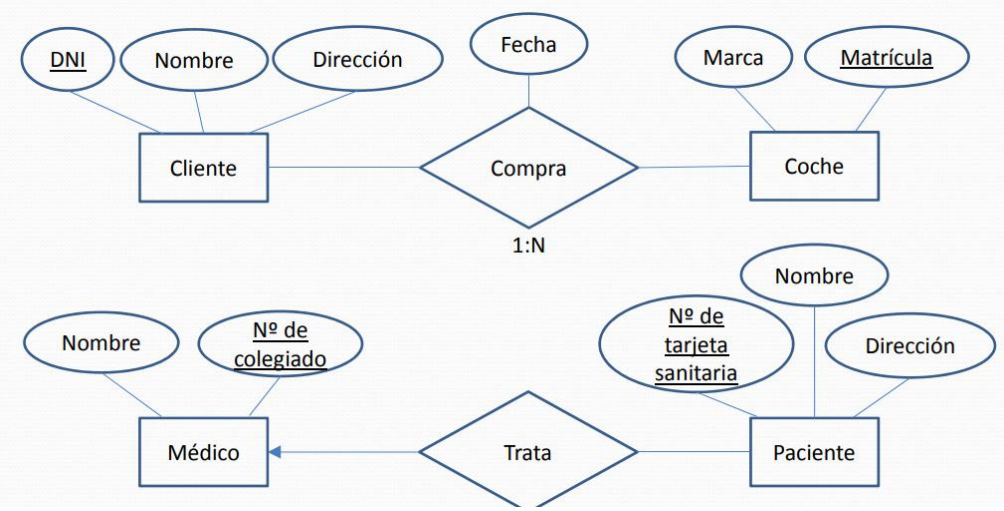
Diagrama entidad-relación

Diagrama entidad-relación: representación grafica de la estructura de una base de datos organizada según el modelo entidad-relación

Sus componentes principales son:

- **Rectángulos**: representan conjuntos de entidades
- **Rombos**: representan conjuntos de relaciones
- **Elipses**: representan atributos (de conjuntos de entidades o de conjuntos de relaciones)
- **Representación de la cardinalidad**. Se puede hacer de varias maneras
    - Con una etiqueta asociada al conjunto de relaciones (solo para relaciones binarias)
    - Poniendo una punta de flecha que señale hacia el conjunto de relaciones que participa de forma “uno” y usando una línea sin punta en el caso de conjuntos de entidades que participan de forma “muchos”.

Ejemplos de diagrama entidad-relación



Modelos lógicos basados en registros: tipos de bases de datos

1. **Modelo de datos jerárquico**

   Permite especificar una base de datos jerárquica, donde se establece una relación jerárquica entre los datos en forma de árbol, y no es posible definir relaciones muchos a muchos

2. **Modelo de datos en red**

   Permite especificar una base de datos en red, donde pueden darse relaciones binarias con cualquier cardinalidad (1:1, 1:N, N:1, N:N) y no es necesario que la estructura tenga forma de árbol

3. **Modelo de datos relacional**

   Permite especificar una base de datos relacional, que estará formada por tablas. Una tabla es una estructura bidimensional formada por una sucesión de registros del mismo tipo.

   Se imponen ciertas condiciones a las tablas, se pueden tratar como relaciones matemáticas.

   Las tablas deben cumplir las siguientes condiciones.

   1. Todos los registros de una tabla son del mismo tipo. Para almacenar registros de tipos distintos se usan tablas distintas
   2. En ninguna tabla aparecen campos repetidos
   3. En ninguna tabla existen registros duplicados
   4. El orden de los registros en la tabla es indiferente. En cada momento se pueden recuperar los registros en un orden particular
   5. En cada tabla hay una llave, formada por uno o varios campos

Transformación del modelo entidad-relación al modelo de datos relacional

Dado un diagrama entidad-relación, el paso a tablas o relaciones del modelo de datos relacional se efectúa como sigue:

- **Conjuntos de entidades**
  - Se define una tabla para cada conjunto de entidades
  - Para cada atributo se define una columna en la tabla
- **Conjuntos de relaciones sin atributos propios**
  - Cardinalidad 1:1: En la tabla de uno de los conjuntos de entidades (el que se considere principal) se añaden las columnas necesarias para albergar los atributos que forman la clave del otro conjunto de entidades
  - Cardinalidad 1:N o N:1: En la tabla del conjunto de entidades que participa de forma "muchos" se añaden las columnas necesarias para albergar los atributos que forman la clave del otro conjunto de entidades
  - Cardinalidad N:M: Se define una tabla propia para el conjunto de relaciones y, en ella, se definen las columnas necesarias para albergar los atributos que forman la clave de cada uno de los conjuntos de entidades que relaciona ese conjunto de relaciones

- **Conjuntos de relaciones con atributos propios**
  - Cardinalidad 1:1, 1:N o N:1: Se puede seguir el mismo enfoque dado para conjuntos de relaciones sin atributos propios, pero añadiendo también las columnas necesarias para albergar los atributos del conjunto de relaciones en la misma tabla del conjunto de entidades donde se añada la clave del otro conjunto de entidades. *Esta forma esta desaconsejada desde el punto de vista conceptual y por cuestiones de mantenimiento del software.*
  - Cardinalidad N:M, 1:1, 1:N o N:1: Se define una tabla propia para el conjunto de relaciones y, en ella, se definen las columnas necesarias para albergar los atributos que forman la clave de cada uno de los conjuntos de entidades que relaciona ese conjunto de relaciones; en esa misma tabla, además, se definen las columnas necesarias para albergar los atributos propios del conjunto de relaciones

Ejemplos

4.4. Sistema de gestión de la base de datos.

Definicion de sistema de gestión de base de datos

Sistema de gestión de base de datos – SGBD ("Data Base Management System" – DBMS): conjunto de software destinado a la creación, control y manipulación de la información de una base de datos.

Un SGBD debe permitir la realización de las siguientes tareas:

1. **Definición** del esquema de la base de datos y de los distintos subesquemas
2. **Acceso** a los datos desde algún lenguaje de alto nivel
3. **Interrogación** (o recuperación de la información) directa en modo conversacional
4. **Organización** física de la base de datos y recuperación tras fallos del sistema

Lenguajes específicos en un SGBD

Las tres primeras tareas se realizan mediante dos lenguajes específicos:

- **Lenguaje descripción de datos (DDL**, "Data Description Language"). Se usa para la descripción del esquema y de los subesquemas
- **Lenguaje de manipulación de datos (DML**, "Data Manipulation Language"). Se utiliza para el acceso a la base de datos desde lenguajes de alto nivel o en modo conversacional

El sistema de gestión de base de datos actúa como intermediario entre los programas de aplicación y el sistema operativo, lo que permite que los programas sean independientes de la estructura física de los datos