domingo, 11 de julio de 2021 11:39 a. m.

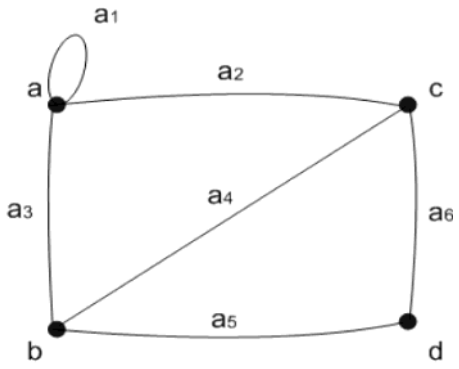
Unidad 1 - Grafos

Concepto

sábado, 19 de junio de 2021 04:54 p. m.

Un grafo puede definirse **como G = (V, A)**, donde **V** representa a un conjunto de puntos, llamados **vértices o nodos**, y **A** es un conjunto de relaciones entre pares de vértices, llamadas **aristas o arcos**.

De esta forma un grafo es un conjunto de vértices y arcos que los relacionan.



Dado que el grafo es un concepto matemático, en lo que se llama Teoría de Grafos Computacional, se utilizaran otros términos para referirse a sus componentes

- Nodos
  - Son los denominados vértices, dado que en un modelo computacional de grafos pueden incluir un conjunto de valores.

- Relaciones
  - Son los identificados aristas o arcos, dado que lo que hacen es relacionar a los nodos.

- Grado
  - El grado de un grafo es la cantidad de arcos que salen de un vértice (grado positivo), o la cantidad de arcos que llegan a un vértice (grado negativo).

Objetivo

sábado, 19 de junio de 2021 05:12 p. m.

Los grafos tienen como **objetivo** fundamental **modelizar un problema específico a través de un modelo abstracto** que establezca elementos que participan en el problema (vértices) y las relaciones que pueden existir entre estos participantes (arcos).

los grafos son estructuras abstractas (una idea de solución), o sea, que no existen realmente sino que **solo sirven como una modelización virtual de un problema real.**

Para su tratamiento computarizado los grafos requerirán de una representación computacional, la cual convertirá esta estructura abstracta en un almacenamiento concreto representado a través de alguna de las representaciones computacionales existentes.

Representación Computacional

sábado, 19 de junio de 2021 05:17 p. m.

En función de la forma y la dinámica que mantengan las representaciones computacionales pueden ser de **2 tipos**:

- **Estática**

  Se construyen sobre **estructuras computacionales rígidas** que utilizan el concepto de **contigüidad** como los ***vectores*** y ***matrices***, o sea que el siguiente vértice este a la derecha y el anterior a la izquierda de un vértice determinado.

  Ocurre lo mismo con los arcos, por eso debe contemplar la existencia de todas las relaciones posibles entre todos los vértices existentes.

- **Dinámica**

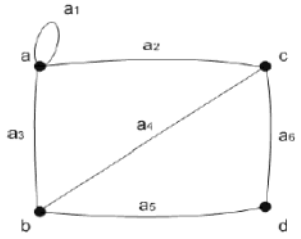
  Se caracterizan por **acompañar la dinámica del grafo**.

  Esto significa que el **espacio utilizado** por la representación **va cambiando** en función de cómo va cambiando el grafo.

Representación Estática

- **Matriz de adyacencia**

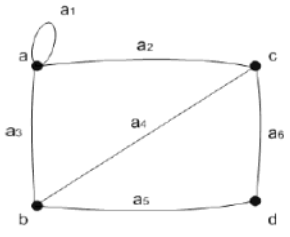
Si consideramos un grafo G = (V, A) con **n vértices** la matriz de adyacencia es aquella de **dimensión** MA_{nxn} con **n como la cantidad de vértices**, donde **la posición** MA_{ij} es el **número de aristas que unen los vértices** $V_i \; y \; V_j$



	a	b	c	d
a	2	1	1	0
b	1	0	1	1
c	1	1	0	1
d	0	1	1	0

- **Matriz de incidencia**

Si se considera un grafo G = (V, A) donde V representa los **n vértices** y A los **m arcos** que componen al grafo, su matriz incidencia es la matriz de **orden nxm**, MI_{nxm} , donde MI_{ij} **es 1 si Vi es incidente con Aj y** MI_{ij} **es 0 en caso contrario**.

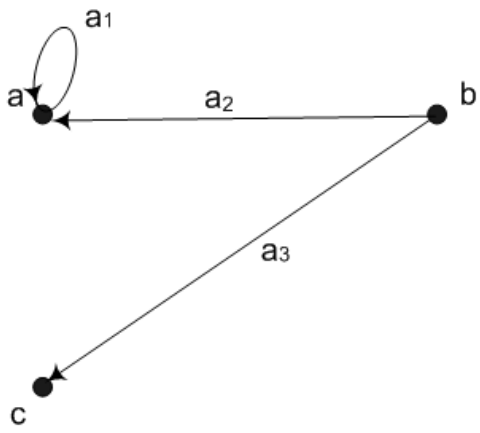


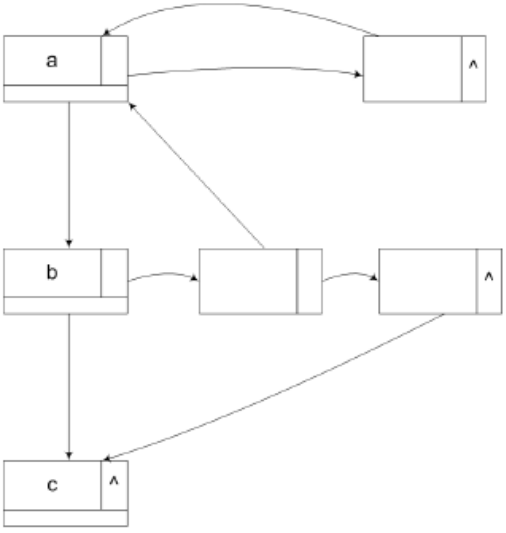
	a_1	a_2	a_3	a_4	a_5	a_6
a	2	1	1	0	0	0
b	0	0	1	1	1	0
c	0	1	0	1	0	1
d	0	0	0	0	1	1

Representación Dinámica

- **Listas de adyacencia**

Las listas de adyacencia es un tipo de representación que se conforma por una **lista que representa los nodos que componen el grafo**, donde **cada una de los elementos** que componen dicha lista de nodos **mantiene otra lista asociada** que **representa los arcos o relaciones que salen de dicho nodo**.





Es importante destacar que **para los grafos no dirigidos el arco está representado dos veces,** una vez en la lista de uno de los vértices que lo conforman y otra vez en el otro vértice, esto se debe a que **todo arco en un grafo no dirigido se lo considera de ida vuelta o sea doble o bidireccional**.

Caracterización

sábado, 19 de junio de 2021 05:43 p. m.

Para todas las caracterizaciones utilizaremos esta definición de grafo

Es aquel grafo que representado como G = (V, A) donde V representa el conjunto de Vértices y A representa el conjunto de arcos.

- **Grafo libre**

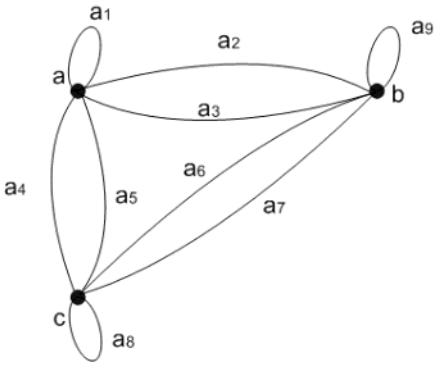
A es un conjunto vacío, no contiene ningún arco (mínima cantidad de relaciones posibles).

De esta forma **un grafo libre es el grafo en el cual no existen arcos**, o sea, que **todos los vértices son aislados.**

- **Grafo completo (Lo opuesto a libre)**

A es un conjunto completo, o sea, que contiene todos los arcos posibles (máxima cantidad de relaciones posibles).

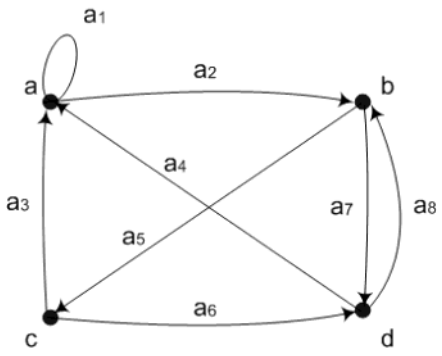
De esta forma **un grafo completo es el grafo en el cual cada vértice está conectado a todos los vértices que componen el grafo, incluido el mismo**.



- **Grafo regular**

Un **grafo es regular** de determinado **grado g**, si cada vértice tiene grado g, o sea, que **todos los vértices tienen el mismo grado g**.

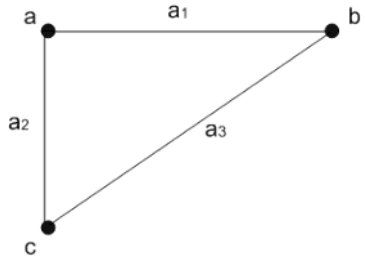
Siempre se considera el grado positivo.



El grado es la cantidad de arcos que salen de un vértice (grado positivo), o la cantidad de arcos que llegan a un vértice (grado negativo).

- **Grafo simple**

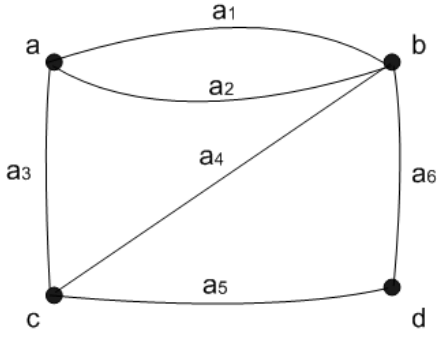
Un **grafo es simple si a lo sumo un arco une dos vértices cualesquiera**, esto es, que existe solo una arista que une a dos vértices específicos.



- **Grafo complejo**

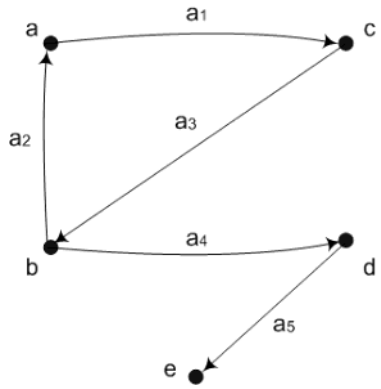
En forma inversa a un grafo simple **un grafo complejo es aquel donde puede existir más de un arco que vincule dos vértices cualesquiera**

Por ello se considera que cualquier grafo que no cumpla con la condición de ser simple se considera complejo.



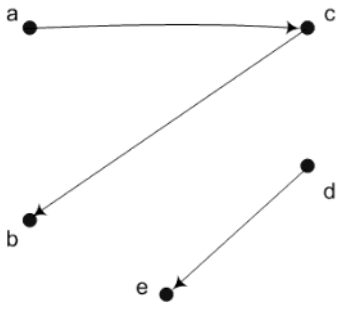
- **Grafo conexo**

  Un grafo se considera conexo si todo par de vértices está conectado por un camino, o sea, si para cualquiera par de vértices existe al menos un camino posible entre ellos, o dicho de otra forma que existe al menos una conexión entre todos los nodos que conforman el grafo, sea esta directa (a través de un arco entre ambos) o indirecta (a través de más de un arco entre ambos).



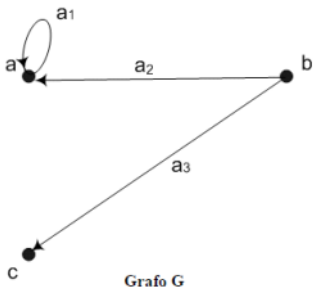
- **Grafo no conexo**

  Se considera **no conexo a un grafo donde un grupo de vértices no esta conectado con el resto de los vértices**, o sea, cualquier grafo que no cumpla con la condición de ser conexo se considera no conexo.

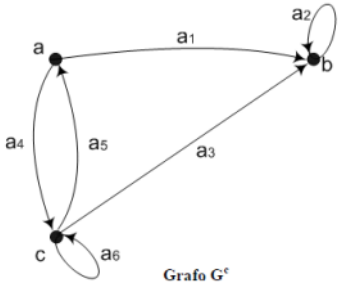


- **Grafo complementario**

  El grafo complementario denominado Gc, es aquel que está compuesto por los mismos vértices que G y el conjunto de aristas son todas aquellas que le faltan a G para ser un grafo completo.



Grafo G



Grafo G^c

Clasificación

sábado, 19 de junio de 2021 06:13 p. m.

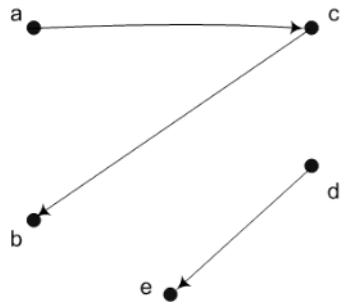
- En función del tipo de relación que implementan hay 2 clasificaciones:

  - **Dirigidos (Utilizados para modelar)**

Son aquellos en los cuales los arcos que vinculan a los vértices tienen una dirección definida, o sea, son arcos con sentido.

Dicho sentido o dirección marca una jerarquía en la relación que se está modelando, donde cada arco tiene un vértice de origen y otro vértice destino de la misma.

Este tipo de relaciones son jerárquicas y resultan ser la mayoría de utilizaciones de los grafos, dentro de estas relaciones encontramos por ejemplo, “ser mayor que”, “ser menor que”, “ser padre de”, “ser componente de”, etc.



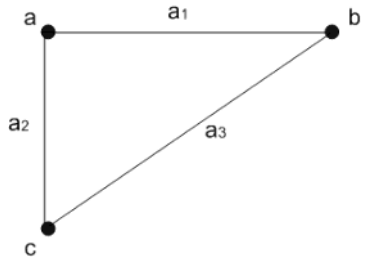
- **No dirigidos**

Son aquellos donde los arcos no tienen una dirección o sentido definido, o sea, donde la relación no establece jerarquía de forma tal que es irrelevante quien es el origen y quien el destino de la relación.

Esta situación solo se da en relaciones simétricas donde el arco realmente representa una relación doble con origen y destino en cada uno de los vértices.

Este tipo de relaciones son las menos utilizadas y dentro de ellas encontramos por ejemplo, “ser igual a”, “ser hermano de”, “ser cónyuge de”, etc.

Como puede observase en todas estas relaciones es irrelevante el orden de evaluación de los vértices que conforman la relación



- En función de las restricciones que pueden ser aplicadas a las relaciones que se modelan existen 2 clasificaciones:

  - **Restrictos**

    Son aquellos grafos en los cuales la **relación** que se modela **NO debe cumplir ninguna de las propiedades de reflexividad, simetría y transitividad simultáneamente. No pueden existir ninguna relación ni reflexiva, ni simétrica, ni transitiva.**

    Los grafos restrictos son más fáciles de administrar y operar, es por ello que los diseñadores que utilizan grafos para modelar problemas tratan de restringir lo más posible las relaciones que se representan a través de ellos.

    Cuanto más acotable el dominio, más fácil será la programación sobre él.

  - **Irrestrictos**

    Son aquellos grafos en los cuales **no se aplica ninguna restricción,** puede modelar relaciones que sean **reflexivas**, **simétricas o transitivas**, también relaciones que cumplan las 3 propiedades simultaneamente.

Caminos y Pasos

sábado, 19 de junio de 2021

- Un **camino** entre dos nodos a y b se establece cuando **existe una vinculación directa o indirecta entre ambos**, esto es cuando se pueden vincular entre sí mediante uno o más arcos, independientemente del sentido de los arcos.

- Un **paso** entre dos nodos a y b se produce cuando **existe un camino entre ambos pero con un sentido preestablecido**, esto es que partiendo del nodo a y siguiendo el sentido de los arcos se llega al nodo b.

    - Como en este caso es relevante el sentido, solo se evalúan pasos en los grafos dirigidos, dado que en los grafos no dirigidos todos los arcos se consideran bidireccionales con lo cual el concepto de paso se iguala al de camino.

- Un **ciclo** entre dos nodos a y b **es un paso o un camino donde el origen y el destino son iguales**, esto es, el vértice de inicio y el vértice de destino son iguales, pudiendo estar compuesto el ciclo por uno o más arcos.

Búsqueda

sábado, 19 de junio de 2021 06:34 p. m.

Para efectuar una búsqueda de un camino o un paso en un grafo existen **dos métodos o técnicas distintas**, que se diferencian en la forma en que realizan el recorrido del grafo para identificar el paso o el camino según corresponda.

- **Búsqueda en profundidad (Depth First)**

Esta técnica se caracteriza por avanzar en profundidad, esto es, sin mantener un orden jerárquico de evaluación, de forma tal la técnica avanza hasta el momento que no puede avanzar más y ahí retrocede para tomar otra relación y seguir avanzando.

Este es un algoritmo recursivo.

- **Búsqueda en anchura (Breath First)**

A diferencia de la búsqueda en profundidad, la búsqueda en anchura, evalúa primero todos los destinos de todos los arcos que parten del vértice origen del paso o camino a evaluar, de forma tal de evaluar primero todos los destinos directos antes de pasar al siguiente.

NINGUNO DE LOS DOS es más rápido que el otro, la rapidez para encontrar lo buscado dependerá de la posición donde este.

Unidad 2 - Estructuras de Datos

Concepto

sábado, 19 de junio de 2021 06:45 p. m.

Una **estructura de datos es un grafo dirigido y restricto**, con las características de unicidad en sus relaciones, esto es que en orden de predecesor, **cada nodo solo puede tener un nodo predecesor a él**.

Dicho de otra manera, hablamos de una estructura de datos cuando a cada nodo solo le llega un arco o flecha.

Es importante destacar que **las estructuras de datos son utilizadas para modelar problemas reales al igual que los grafos**; la ventaja comparativa es que debido a las limitaciones de las mismas por ser grafos restrictos unívocos, se simplifica su administración.

También es fundamental la utilización de estas estructuras como un medio de simplificar la programación en función de dar soporte a algoritmos utilizando las diferentes características de cada una de ellas de forma tal de disminuir la tarea de programación evitando realizar engorrosos algoritmos que resuelvan un problema determinado.

El sentido de las relaciones es inverso al orden en el que se salen los elementos de la estructura analizada.

Clasificación

sábado, 19 de junio de 2021 06:55 p. m.

Las estructuras de datos se pueden dividir inicialmente en:

- **Biunívocas**

Se caracterizan por ser univocas en ambos sentidos de la relación manteniendo uno o ningún sucesor.

Dentro de ellas encontramos las **pilas, las colas y las listas.**

- **Univocas**

Se caracterizan por ser univocas en un sentido de la relación, manteniendo un solo predecesor pero pudiendo tener más de un sucesor.

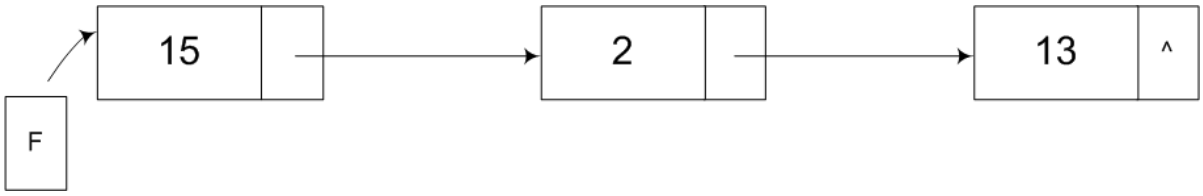
Dentro de ellas encontramos a los **árboles.**

Listas

sábado, 19 de junio de 2021 07:03 p. m.

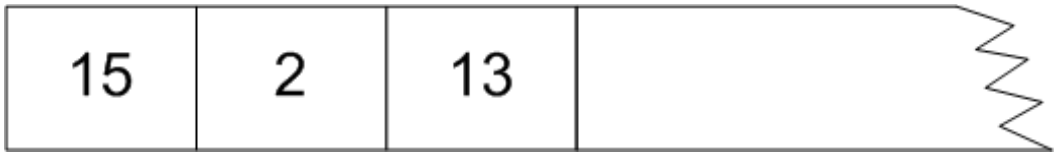
Estructura de datos que tiene una **dinámica abierta**, esto es que dentro de una lista a la hora de realizar un alta se recorre toda la lista y se coloca el elemento a insertar en la posición que se requiera, esto dependerá de que se desee o no mantener la lista ordenada por algún valor de los nodos que la componen.

Representación dinámica de una lista



Esta figura nos muestra una forma de implementar una lista sobre una representación dinámica de forma tal que solo se utiliza el espacio ocupado por los nodos que existen en un momento determinado.

Representación estática de una lista



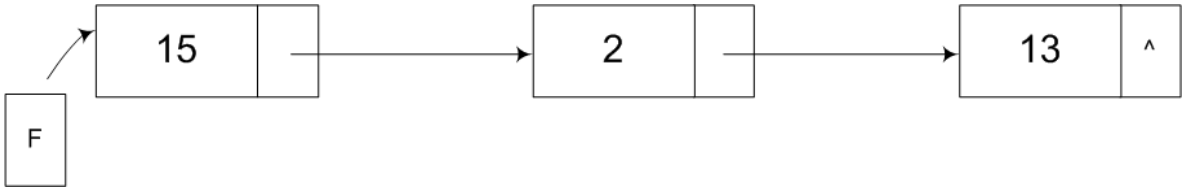
En esta representación la lista se representa sobre un vector, el cual se define de una dimensión predeterminada y se va completando en función de los elementos que se ingresan a la estructura.

Tipos de Listas

Una lista normalmente tiene un principio y un final, pero en función de la forma
que puede tomar en su presentación encontramos tres tipos diferentes de listas estas son:

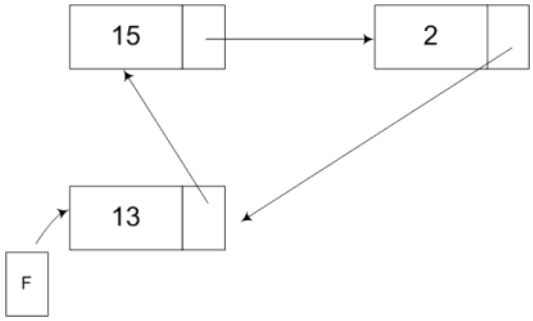
- **Lista lineal**

Es la lista tradicional, o sea, aquella que comienza con un elemento y en la cual el último puntero del nodo apunta a NULL.



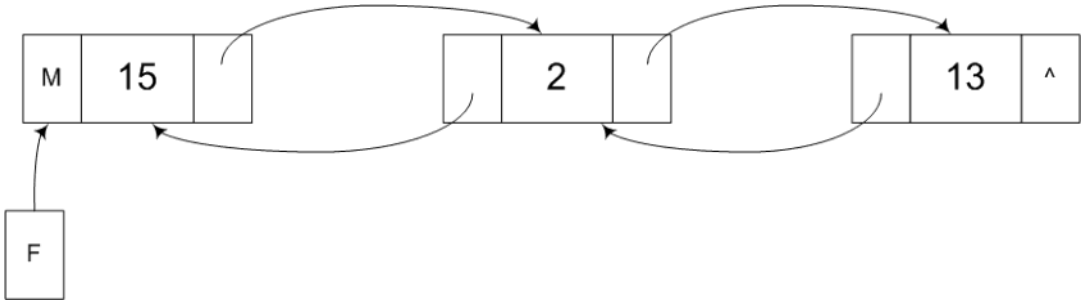
- **Lista circular**

Es la lista en la cual el último nodo no lleva un apuntador en NULL, sino que apunta al primero.



- **Lista doblemente enlazada**

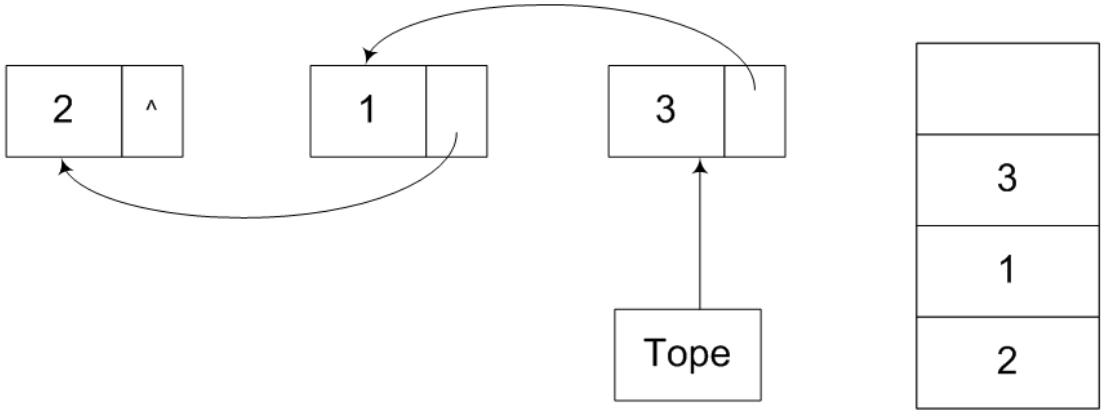
Son las listas que se implementan con la posibilidad de que los nodos, aparte de tener un apuntador al nodo siguiente, tengan un apuntador al nodo anterior.



Pila

sábado, 19 de junio de 2021 07:12 p. m.

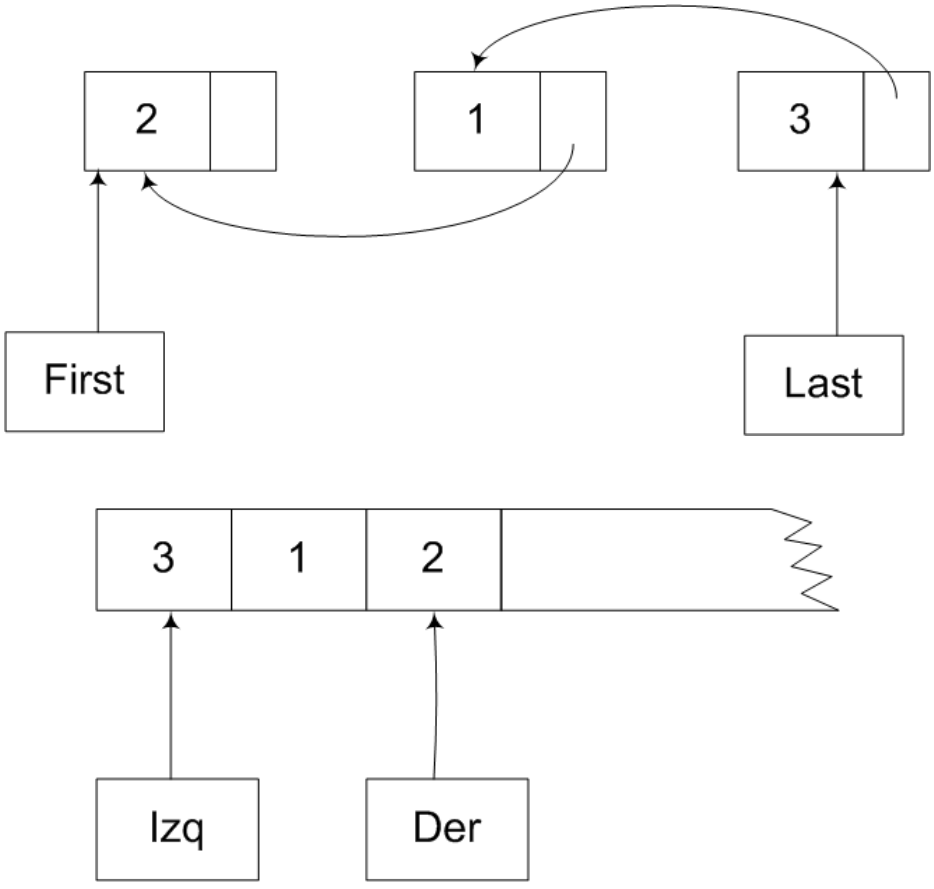
Estructura de datos que tiene como característica diferencial que s**u dinámica de ingreso y egreso es de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir)** de forma tal que la forma de ingresar los datos es por un extremo de la pila y por el mismo extremo se realizan las extracciones de la misma.



Cola

sábado, 19 de junio de 2021 07:19 p. m.

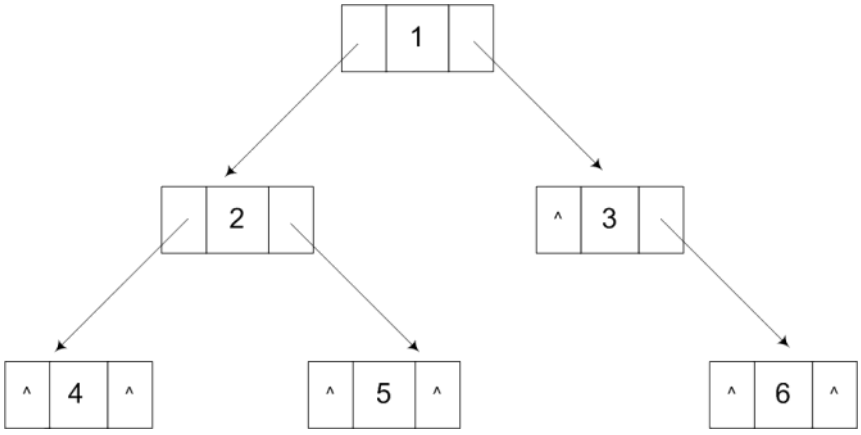
Estructura de datos que se caracteriza por **privilegiar el orden y la jerarquía en una estructura de datos, manteniendo una dinámica que se la conoce como FIFO (del inglés First In First Out)**, debido a que el primer elemento en entrar será también el primero en salir respetando el orden de llegada.



Árboles

sábado, 19 de junio de 2021 07:22 p. m.

Este tipo de estructura tiene la característica de no ser biunívoco, dado que solo cumple la unicidad en un sentido sabiendo que cada elemento tiene un solo predecesor pero que puede tener más de un sucesor.



Lo desarrollaremos en la Unidad 3.

domingo, 11 de julio de 2021 11:45 a. m.

Unidad 3 - Árboles

Conceptos

sábado, 19 de junio de 2021 07:30 p. m.

- **GRADO**

Máxima cantidad de hijos o subárboles que puede tener cada nodo, es el posible crecimiento del árbol.

- **NIVEL**

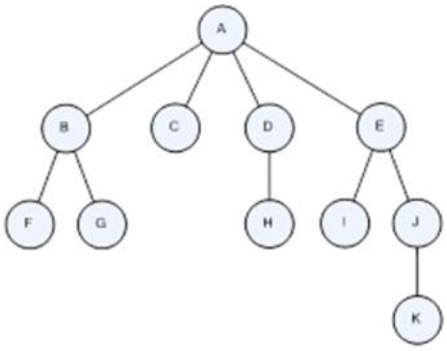
Posición en la que se encuentra cada nodo con respecto a la raíz del mismo, considerando que la raíz se encuentra en el nivel 0

- **PRODUNDIDAD**

Cantidad de niveles con que cuenta el mismo.

A cuantos pasos de la raíz estará el elemento más profundo del árbol.

Ejemplo



- Grado: 2

- Niveles

0) A
1) B - C - D - E
2) F - G - H - I - J
3) K

- Profundidad: 3 niveles (0,1,2,3)

Representación Computacional

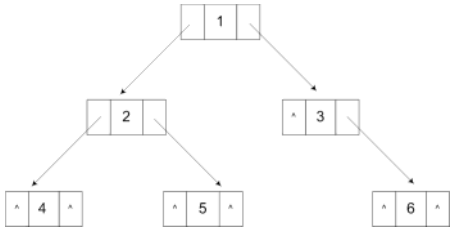
sábado, 19 de junio de 2021 07:55 p. m.

ESTATICA

- Un árbol se representa en forma estática a través de un vector.

Se guarda por orden convencional (del primer nivel al último y de izquierda a derecha del nivel).

Árbol almacenado:



<u>Para saber la posición de un hijo de cierto elemento en el vector (caso árbol binario)</u>

N: posición de elemento padre en el vector

2N + 1 (Para hijo izquierdo) 2N + 2 (Para hijo derecho)

Para un caso genérico podríamos pensarlo como:

Grado . N + posición dentro del nivel (siempre arranca en 1)

Ej: Para el nodo padre 2 (su posición en el vector es 1)

Posición Hijo izquierdo = 2 . 1 + 1 = 3

Posición Hijo derecho = 2 . 1 + 2 = 4

<u>Para saber la posición del padre de cierto elemento en el vector (caso árbol binario)</u>

H: posición de elemento hijo en el vector

(H - 1) / 2 (Para hijo izquierdo) (H - 2) / 2 (Para hijo derecho)

Para un caso genérico podríamos pensarlo como:

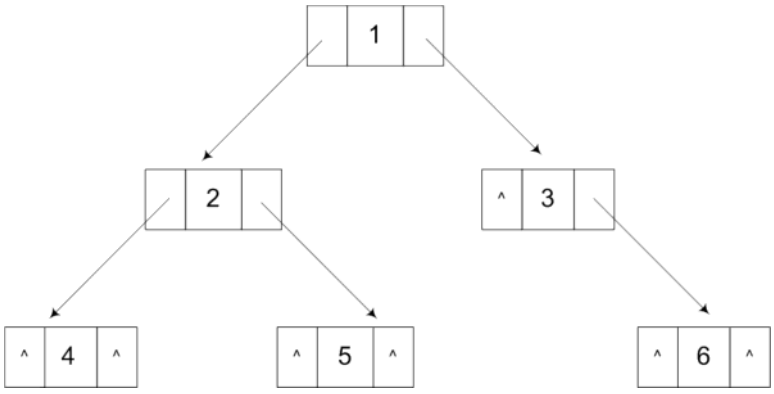
(H - posición del nodo hijo dentro del nivel) / Grado

Ej: Para el nodo hijo 5 (su posición en el vector es 4)

Posición padre = (4 - 2) / 2 = 1

DINAMICA

- Un árbol se representa en forma dinámica a través de un conjunto de nodos de igual tipo vinculados entre si a través de punteros o links, cada elemento tendrá n cantidad de punteros para sus sucesores (grado del árbol)



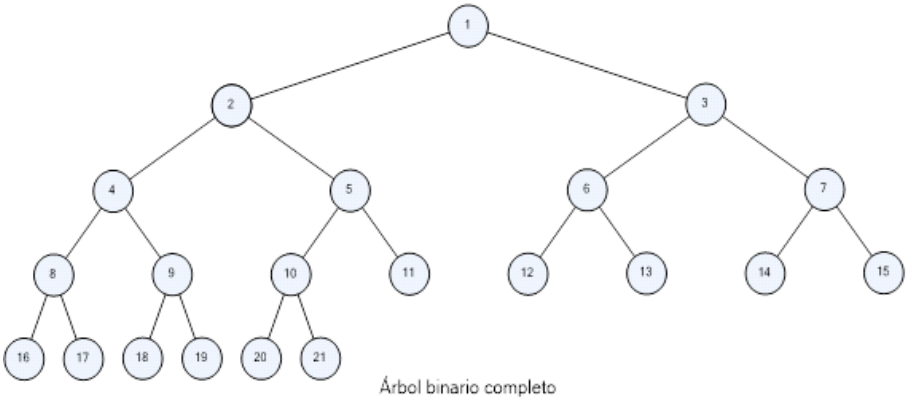
Características

sábado, 19 de junio de 2021 08:07 p. m.

En los árboles se analizan características que no se analizan en otras estructuras de datos debido a que las restantes son biunívocas.

COMPLETO

- Un árbol completo es aquel en el cual todos los nodos del árbol cumplen el grado o son hojas.



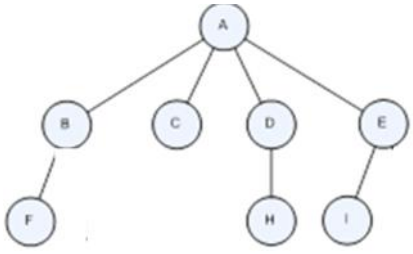
Árbol binario completo

BALANCEADO

- Un árbol está balanceado si todos los subárboles desde la raíz pesan los mismo o sea tienen la misma cantidad de elementos o una diferencia indivisible entre los subárboles

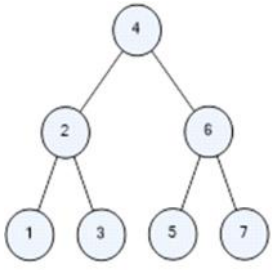
  nota: (diferencia que sea indivisible por el grado del árbol) definir esto mejor

Ejemplo de árbol balanceado:



PERFECTAMENTE BALANCEADO

- Un árbol está perfectamente balanceado, cuando está balanceado en todos sus niveles



Árbol binario perfecto

Crecimiento

sábado, 19 de junio de 2021 08:26 p. m.

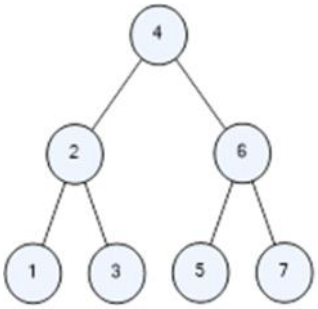
El crecimiento de un árbol **es exponencial en función del grado del mismo**, o sea, que en cada nivel puede crecer en función del grado definido.

La máxima cantidad de elementos posibles de un árbol está definida en función de la siguiente fórmula:

$$\boldsymbol{Max\ Elementos = Grado^{niveles} - 1}$$

Para que se cumpla esto el árbol debe estar completo y perfectamente balanceado.

De esta forma si tomamos el árbol de la figura veremos que la máxima cantidad de elementos que puede tener es 7 que responde a 2 que es el grado elevado a 3 que son los niveles menos 1 que está dado por el grado de imparidad de la raíz.



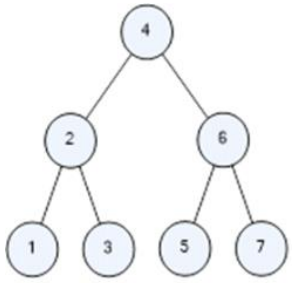
Árbol binario perfecto

Búsqueda

sábado, 19 de junio de 2021 08:28 p. m.

De la misma forma si se quisiera encontrar un elemento en dicho árbol, **la búsqueda se realiza por niveles y no por elementos** de forma tal que **para encontrar cualquier elemento a lo sumo realizaremos tantas preguntas como niveles tenga el árbol**.

En este caso 3.



Árbol binario perfecto

Si tomamos la fórmula de crecimiento y despejamos de ella la cantidad de niveles llegamos a la siguiente fórmula de búsqueda que justifica que **un árbol tiene una búsqueda logarítmica.**

<u>Si el árbol no es perfectamente balanceado su búsqueda tenderá a ser logarítmica.</u>

elementos = $grado^{niveles}$ - 1

elementos + 1 = $grado^{niveles}$

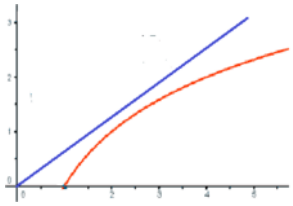
log elementos + 1 = niveles * log grado

niveles = log elementos + 1 – grado

niveles > log elementos

- **log elementos** representa las búsquedas que debería realizar el árbol para encontrar un elemento, ya que no siempre se analiza la cantidad de veces indicada por el nivel del árbol (esto ocurre cuando se busca un nodo hoja), a veces se realizan menos comparaciones por ejemplo si se buscara la raiz.

Para poca cantidad de elementos no hay mucho desvio entre una búsqueda lineal y logarítmica, pero para una gran cantidad de elementos la logarítmica resultará mucho más rápida que una lineal.



Una estructura de datos básica como por ejemplo una lista tiene una búsqueda secuencial, por lo tanto, si tengo 10 elementos a lo sumo debería realizar 10 preguntas para encontrar un elemento.

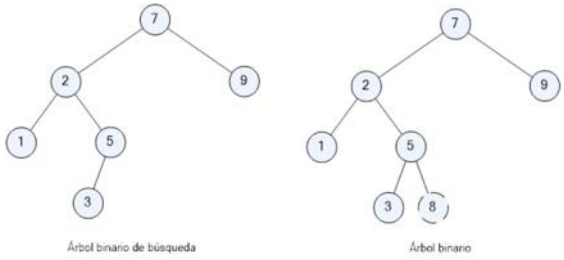
En un árbol esa búsqueda no es lineal, sino que es logarítmica.

Árbol Binario de Búsqueda

sábado, 19 de junio de 2021 08:48 p. m.

Un ABB como lo indica su nombre es un árbol de grado 2 diseñado para buscar como método alternativo a una lista representada en un vector, donde los elementos menores a la raíz se ingresan a la izquierda y los elementos mayores a la raíz se ingresan a la derecha.

Siempre es más rápida la búsqueda utilizando arboles binarios de búsqueda en comparación a las listas, excepto el caso en que todos los elementos del árbol se encuentren almacenados linealmente en una rama.



Ejemplo

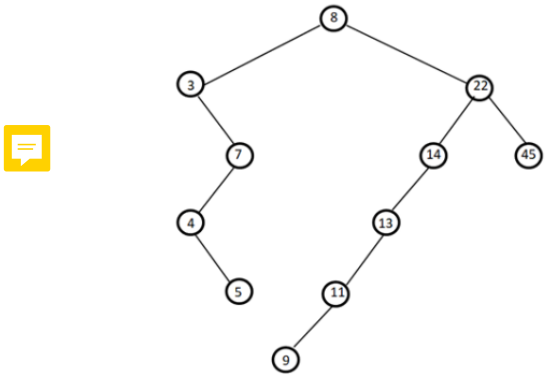
Dado el siguiente conjunto de números desordenados:

8 – 22 – 14 – 13 – 45 – 11 – 3 – 7 – 4 – 5 - 9

Si utilizamos una lista para ordenar los mismos, por ejemplo implementada en un vector nos quedaría lo siguiente:

3	4	5	7	8	9	11	13	14	22	45

Si utilizamos un ABB nos quedaría lo siguiente:



- **1ero**: al entrar el 8 toma el lugar de raíz
- **2do**: al entrar el 22 compara con el 8 y lo ubica a la derecha ya que es mayor al 8
- **3ero**: al entrar el 14 compara con el 8 y como es mayor analiza el nodo derecho del 22, compara el 14 con el 22, como 14 es menor a 22 lo ubica como hijo izquierdo del 22
- **4to:** entra el 13 y deriva por el árbol comparando hasta ubicarlo como hijo izquierdo del 14, ya que 13 es mayor que 8 y menor que 22 y menor que 14
- **5to:** entra el 3 y al comparar con 8 deriva por la izquierda y toma posición como nodo hijo izquierdo de la raíz ya que el numero 3 < 8

Si se quisiera mejorar la búsqueda en este árbol se debería acomodar la raíz para que sea lo más balanceado posible y que tenga menos niveles, lo cual disminuirá la cantidad de comparaciones a realizar y por lo tanto disminuyen la cantidad de comparaciones necesarias para encontrar un elemento.

Barrido

sábado, 19 de junio de 2021 08:53 p. m.

El **Barrido** representa la lectura del árbol.

Un **BARRIDO de un árbol ES LA FORMA DE LEER EL MISMO**, si bien existe una forma de recorrer un árbol que es de arriba hacia abajo y de izquierda a derecha por convención occidental, existen tres formas distintas de leer los elementos que conforman tres barridos diferentes:

- **PRE**ORDEN

El nodo se lee apenas se llega al mismo

- **POST**ORDEN

El nodo se lee cuando se va del mismo y no se va a regresar

- **IN**ORDEN

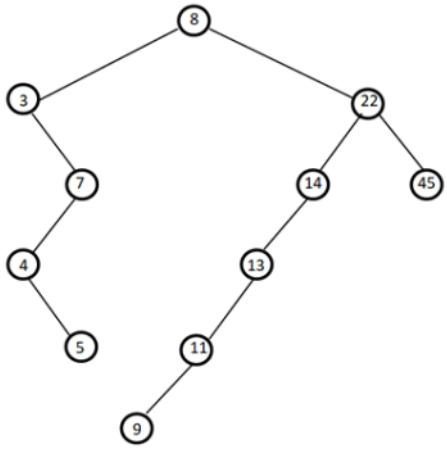
El nodo se lee cuando se cambia de rama en el árbol, esto ocurre cuando se evalúa la rama "izquierda" del nodo y al momento de leer la rama "derecha" del nodo se lee el valor que almacena ese nodo.

Si al evaluar la rama "izquierda" debe avanzar en profundidad no se lee el valor del nodo, se leerá cuando se retorne y se deba evaluar la rama "derecha".

ORDEN referencia a que es el orden normal, y el **PREFIJO** es cuando hago la lectura

Ejemplo: El preorden es leerlo previo a aplicar el orden convencional

Dado el siguiente árbol veremos cuál es el resultado de cada barrido



Preorden: 8-3-7-4-5-22-14-13-11-9-45

Postorden: 5-4-7-3-9-11-13-14-45-22-8

Inorden: 3-4-5-7-8-9-11-13-14-22-45

Si barremos **INORDEN** un **ABB** obtendremos la lista de elementos ordenada, porque siempre leeremos primero todo lo de la rama izquierda (todo lo menor a la raíz), luego la raíz y por ultimo todo lo de la rama derecha (todo lo mayor a la raíz).

El algoritmo de barridos es siempre igual, dado que un árbol se basa en la recursividad, lo único que se modifica es el momento en que se lee el nodo.

PREORDEN

```
if root
        printf(root->dato);
        preorden(root->izq);
        preorden(root->der);
return;
```

INORDEN

```
if root
        inorden(root->izq);
        printf(root->dato);
        inorden(root->der);
return;
```

Árbol de Expresión

sábado, 19 de junio de 2021 09:09 p. m.

Describir el árbol de expresión.

Un arbol de expresion es aquel arbol que representa una expresion matematica, permite recorrerla y, eventualmente, resolverla. Se construye de la siguiente forma por ej. para la expresion [(5-4) * 2]:

-primero se pasa la expresion a representar en el arbol a notacion polaca : * (- 5 4) 7 . Los valores siempre seran hojas del arbol mientras los operadores nunca lo seran.

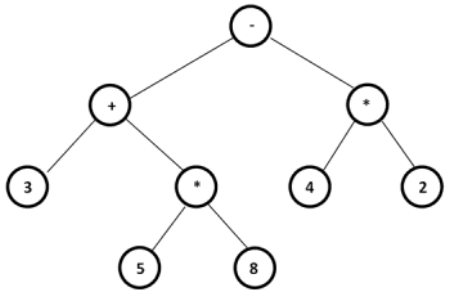
-se empieza a construir el arbol agarrando como raiz al primer operador (*). Luego a izquierda se ubica el operador (-), a izquierda del (-) se asigna el 5 y a derecha del (-) el 4. Por ultimo a derecha de la raiz (*) se ubica el 7.

Ya construido el arbol se lo puede recorrer en Inorden obteniendo la notacion matematica infija, PostOrden obteniendo notacion polaca inversa o en PreOrden.

Una expresión puede representarse y resolverse a partir de un árbol, a estos se los llama árboles de expresión.

3 + 5 * 8 – 4 * 2

POLACA INVERSA: - + 3 * 5 8 * 4 2



INORDEN: 3 + 5 * 8 – 4 * 2

POSTORDEN: 358*+42*-

Si a dicho árbol se barre **INORDEN** se obtiene la expresión matemática en **notación INFIJO**, si se lo barre **POSTORDEN** se obtiene la expresión matemática en **notación POSTFIJO o POLACA INVERSA**.

Actualmente esto de las operaciones funciona con 2 pilas, una de operandos y otra de operadores.

domingo, 11 de julio de 2021 11:45 a. m.

Unidad 4 - Métodos de Clasificación

Concepto

lunes, 28 de junio de 2021 06:50 p. m.

El objetivo es dado un conjunto de valores desordenados del tipo {a1, a2,…, an}, devolver un conjunto ordenado de menor a mayor o de mayor a menor.

Por ejemplo una secuencia de entrada como:

- {31, 41, 59, 26, 41, 58} retorna la secuencia {26, 31, 41, 41, 58, 59}

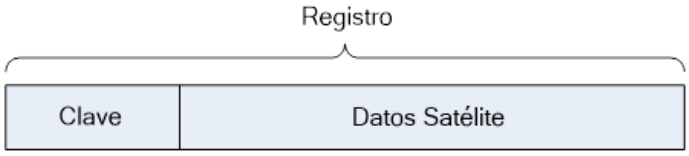
Técnicamente **son métodos de Clasificación** porque en la Computación se le llama **clasificar** a **ordenar en un momento determinado**, a diferencia de lo que se le llama **ordenar** que es **crear una estructura que permita mantener ordenado los datos a través del tiempo.**

Registros

lunes, 28 de junio de 2021 07:01 p. m.

En la práctica muy pocas veces los números a ordenar son valores aislados, cada número suele ser parte de una colección de datos llamado registro.

Cada registro contiene una clave (key), que es el valor a ser ordenado, y el resto del registro contiene los datos satélites.



El hecho de ordenar simples números o registros no es relevante para el método de clasificación en sí, por lo que generalmente se asume que la entrada simplemente consiste en números o letras.

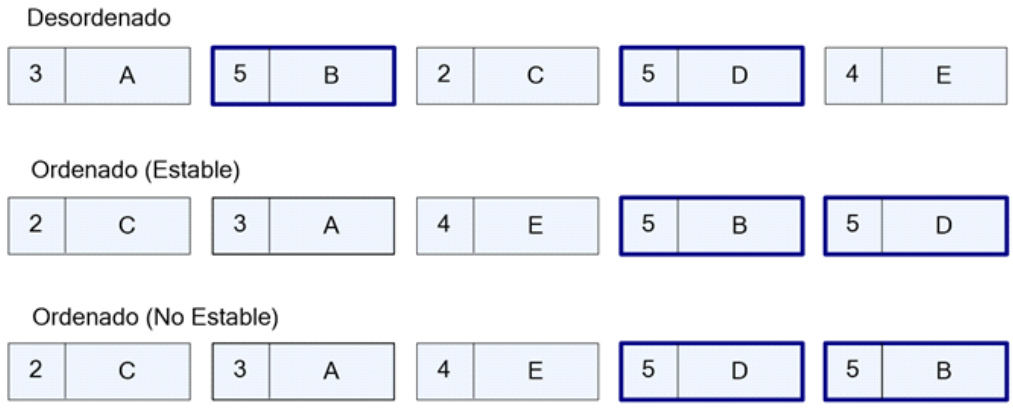
Estabilidad

lunes, 28 de junio de 2021 07:03 p. m.

Un **ordenamiento** se considera **estable si mantiene el orden relativo que tenían originalmente los elementos con claves iguales**, o sea si entra un elemento con el mismo "valor" que otro elemento se respeta el orden en el que entraron a la estructura para ordenarlos.

Por eso (5,B) debe quedar antes del (5,D).

Si se tiene dos registros A y B con la misma clave en la cual A aparece primero que B, entonces el método se considera estable cuando A aparece primero que B en el archivo ordenado.



In Situ

lunes, 28 de junio de 2021 07:08 p. m.

In Situ (en el lugar, en la situación en la que estoy), refiere a que **el algoritmo no utiliza estructuras auxiliares para realizar el ordenamiento**.

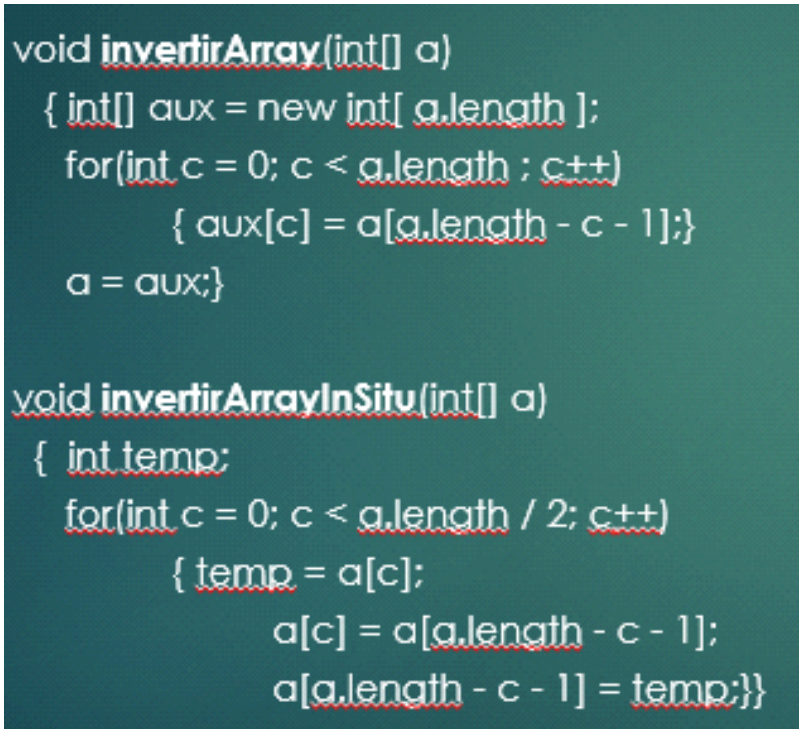
Los métodos **in situ** son los que **transforman una estructura de datos usando el mismo espacio ocupado originalmente o en algunos casos una cantidad extra de memoria**, siendo ésta pequeña y constante.

Generalmente **la entrada es sobrescrita por la salida a medida que se ejecuta el algoritmo**.

Por el contrario los algoritmos que no son in situ requieren gran cantidad de memoria extra para transformar una entrada.

Esta característica es **fundamental en lo que respecta a la optimización de algoritmos**, debido a que el hecho de **utilizar la misma estructura disminuye los tiempos de ejecución**, debido a que no se debe utilizar tiempo en crear nuevas estructuras, ni copiar elementos de un lugar a otro.

Ejemplos



```
void invertirArray(int[] a)
  { int[] aux = new int[ a.length ];
    for(int c = 0; c < a.length ; c++)
          { aux[c] = a[a.length - c - 1];}
    a = aux;}

void invertirArrayInSitu(int[] a)
 {  int temp;
    for(int c = 0; c < a.length / 2; c++)
          { temp = a[c];
                a[c] = a[a.length - c - 1];
                a[a.length - c - 1] = temp;}}
```

Clasificación - Interna y Externa

lunes, 28 de junio de 2021 07:16 p. m.

- **MÉTODO INTERNO**
  - Si el archivo a ordenar cabe en memoria principal (por ejemplo la RAM), entonces el método de clasificación es llamado método interno.

- **MÉTODO EXTERNO**
  - Si ordenamos archivos desde un disco u otro dispositivo óptico que no es memoria, se llama método de clasificación externo.

Hay algoritmos **aptos para clasificación interna** y **otros para clasificación externa**, eso **dependerá** de la **optimización** y de la "**complejidad computacional**" del **algoritmo**.

Complejidad - Concepto

lunes, 28 de junio de 2021 07:21 p. m.

La **complejidad computacional** mide **que tan complejo** es para la computadora **ejecutar un determinado algoritmo**.

Cuanto más complejo es lo que tenga hacer la máquina para ejecutar el algoritmo, más le costará realizarlo (por ejemplo: tiempo o recursos) de esto surge el concepto de **costo computacional**.

- **Complejidad "P" (Posible de resolución)**

Esto significa que la **complejidad computacional del algoritmo es aceptable** derivando a un **costo computacional también aceptable**.

Es el conjunto de los problemas de decisión que pueden ser resueltos en una máquina determinista en tiempo a lo sumo polinómico, lo que corresponde intuitivamente a problemas que pueden ser resueltos aún en el peor de sus casos.

- **Complejidad "NP" (No Posible de resolución)**

Esto no significa que NO exista algoritmo que resuelva el problema, sino que **su complejidad computacional es inaceptable**, provocando que su costo computacional sea muy elevado.

Es el conjunto de los problemas de decisión que pueden ser resueltos por una máquina no determinista en tiempo mayor que polinómico.

Evaluar Complejidad

lunes, 28 de junio de 2021 08:35 p. m.

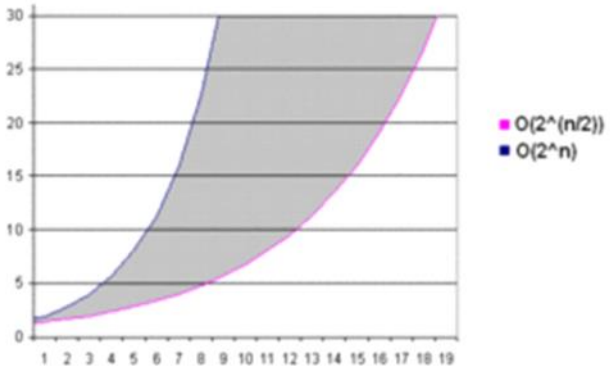
Como evaluar la Complejidad de un Algoritmo

- Las computadoras no tienen discernimiento, solo puede realizar operaciones matemáticas y comparaciones por su característica electrónica booleana.

- Para evaluar la complejidad en un algoritmo determinado se evalúan principalmente la cantidad de comparaciones realizadas, ya que una comparación (if) puede llegar a ser hasta 100 veces más lenta que una operación matemática básica.

Para **determinar la complejidad de un algoritmo** se observa cuál es el comportamiento del mismo, y **en función de la cantidad de comparaciones** que haga va a encontrar una función de orden de complejidad que la represente.

El **orden de complejidad** se describe como **O(función)** donde función es la **función matemática que acota el comportamiento del algoritmo en función del tiempo y la cantidad de elementos.**



Eje x: cantidad de elementos de entrada
Eje y: tiempo de ejecución del algoritmo

Teoría de la complejidad

La teoría de la complejidad trata de **encontrar la función matemática que representa la ejecución de un algoritmo** para saber si ese algoritmo forma parte del **conjunto P o el conjunto NP**.

- La teoría de la complejidad mide la factibilidad técnica de un algoritmo.

Esto va a depender de la proporcionalidad de la función matemática en cuestión:

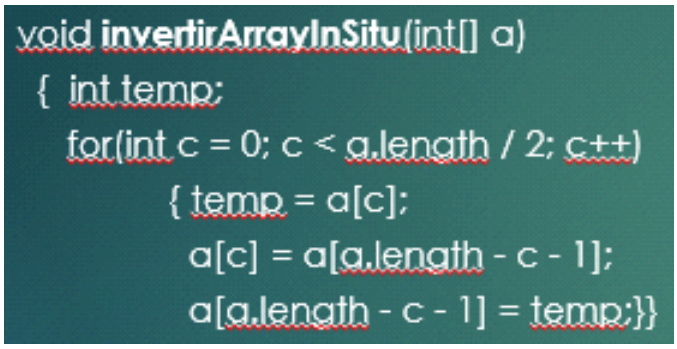
- **Si es proporcional o inferior a la proporcionalidad** (linealidad) entonces el **algoritmo forma parte del conjunto P**.

- **Si supera la proporcionalidad** (linealidad) entonces se considera que el **algoritmo es parte de lo que se llama el conjunto NP**.

Cuando se desarrolla un algoritmo profesional **NO** puede superar la linealidad, como mínimo debe ser proporcional.

Complejidad - Ejemplo

lunes, 28 de junio de 2021 08:53 p. m.



- Este algoritmo si bien realiza asignaciones, el orden de complejidad se establece a partir de analizar que función matemática acota la cantidad de comparaciones que realiza en función de los elementos.

- De esta forma vemos que este algoritmo realiza n/2 comparaciones, si tomamos como n la cantidad de elementos del array, en función de ello decimos que su orden de complejidad es O(n/2).

Algoritmos

lunes, 28 de junio de 2021 09:08 p. m.

Existe una tendencia en pensar que un algoritmo es mejor que otro, pero no es así.

Por ejemplo en el ORDER BY no utiliza solo uno, sino que tiene muchos de estos algoritmos programados los cuales puede utilizar. **El algoritmo de clasificación a utilizar DEPENDE del contexto en el que se apliquen**.

Los algoritmos de clasificación infieren (concluir a partir de las condiciones del contexto) cual será el mejor algoritmo a ser aplicado para un contexto determinado.

Por ejemplo se podría tender a pensar que Bubble Sort es el peor algoritmo de clasificación, pero si los datos ya se encuentran previamente ordenados este será el mejor algoritmo a aplicar, ya que su orden de complejidad sería O(n).

Nombre	Mejor caso	Caso medio	Peor caso	Estable	Comentarios
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	Si	El más lento de todos. Uso pedagógico.
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Si	Apto si queremos que consumir siempre la misma cantidad de tiempo.
Insertion Sort	$O(n)$	$O(n)$	$O(n^2)$	Si	Conveniente cuando el array esta casi ordenado.
Shell Sort	$O(n^{5/4})$	$O(n^{3/2})$	$O(n^2)$	No	Dependiente de la secuencia de incrementos.
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Si	Adecuado para trabajos en paralelo.
Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	No	El método acotado en el tiempo muy utilizado para grandes volúmenes de datos
QuickSort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	No	El más rápido en la práctica. Implementado en gran cantidad de sistemas.

Bubble Sort

lunes, 28 de junio de 2021 08:55 p. m.

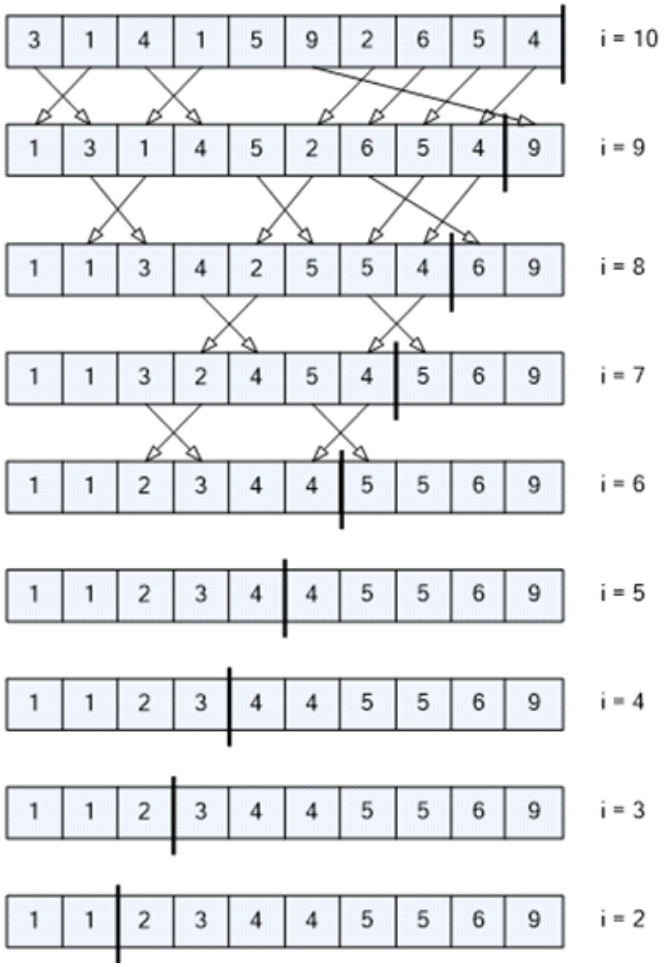
Bubble Sort, o método de la burbuja o de intercambio directo es **uno de los métodos más simples y elementales**, pero **también uno de los más lentos y poco recomendables**.

- Consiste en hacer pasadas sobre los datos, donde en cada paso, los elementos adyacentes son comparados e intercambiados si es necesario.

- A veces no hace falta hacer pasadas sobre el array para que éste quede ordenado, sino que puede quedar ordenado antes de terminar todas las pasadas.

- El **tiempo de ejecución** del Bubble Sort en el peor de los casos es de **O(n2)**, y ocurre cuando el array viene en orden inverso.

- Sin embargo hay un caso en el que el Bubble Sort puede ordenar en tiempo lineal, y es cuando el array esta previamente ordenado, resultando en un tiempo de ejecución de O(n).



Selection Sort

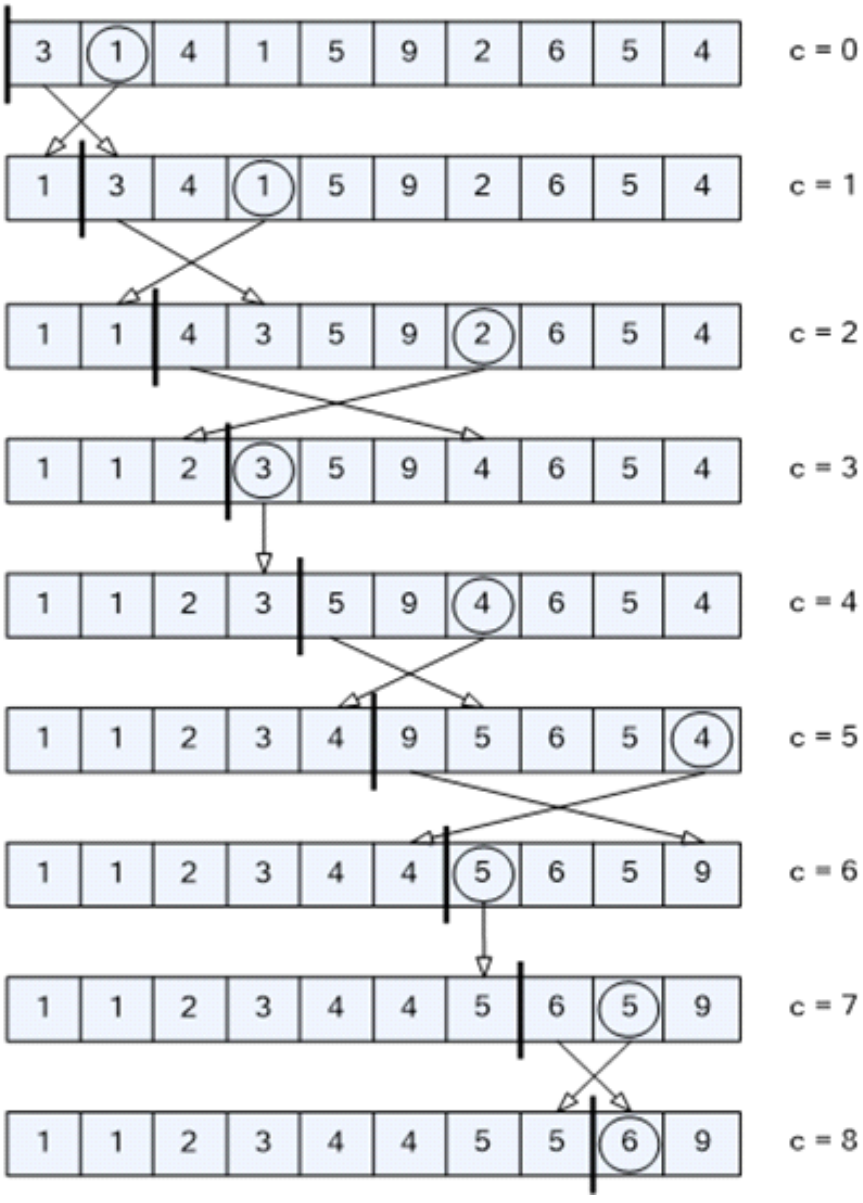
lunes, 28 de junio de 2021 09:03 p. m.

- Selection Sort u Clasificación por Selección, es otro de los métodos elementales, que es necesario conocer a fin de tratar luego los más complejos.

- Comienza buscando el elemento más pequeño del array y se lo intercambia con el que está en la primera posición, luego se busca el segundo elemento más pequeño y se lo coloca en la segunda posición. Se continua con este proceso hasta que todo el array este ordenado.

- Debido a que la mayoría de los elementos se mueven a lo sumo una vez, resulta muy bueno para ordenar archivos que tienen registros muy grandes y claves muy pequeñas.

- A diferencia del Bubble Sort no tiene corte anticipado y **su orden de complejidad para el peor caso es O(n2)**.



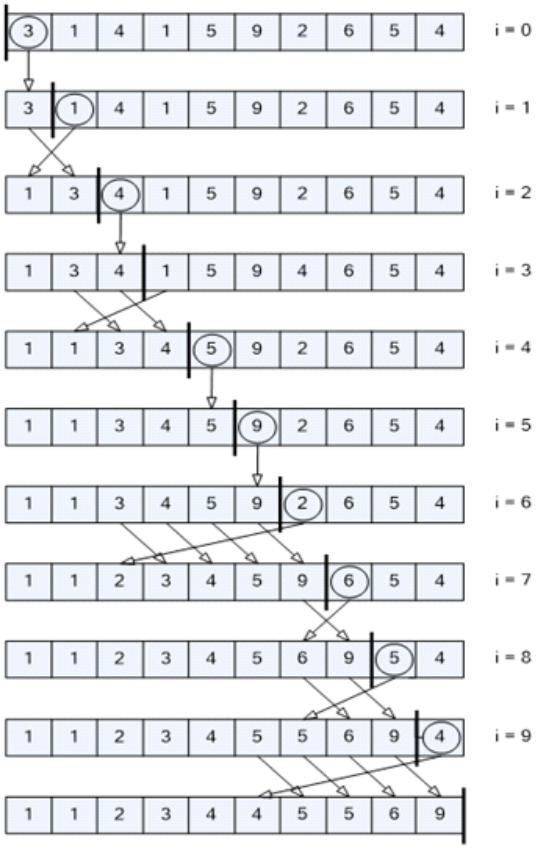
Insertion Sort

lunes, 28 de junio de 2021 09:05 p. m.

- Este método se basa en la idea del ordenamiento parcial, en el cual hay un marcador que apunta a una posición donde a su izquierda se considera que están los elementos parcialmente ordenados, es decir ordenados entre ellos pero no necesariamente en sus posiciones finales.

- El algoritmo comienza eligiendo el elemento marcado para poder insertarlo en su lugar apropiado en el grupo parcialmente ordenado, para eso sacamos temporalmente al elemento marcado y movemos los restantes elementos hacia la derecha. Nos detenemos cuando el elemento a ser cambiado es más pequeño que el elemento marcado, entonces ahí se intercambian el elemento que esta en esa posición con la del elemento marcado.

- El **tiempo de ejecución es O(n2)** y es alcanzable si el array viene ordenado en orden inverso.



Shell Sort

lunes, 28 de junio de 2021 09:10 p. m.

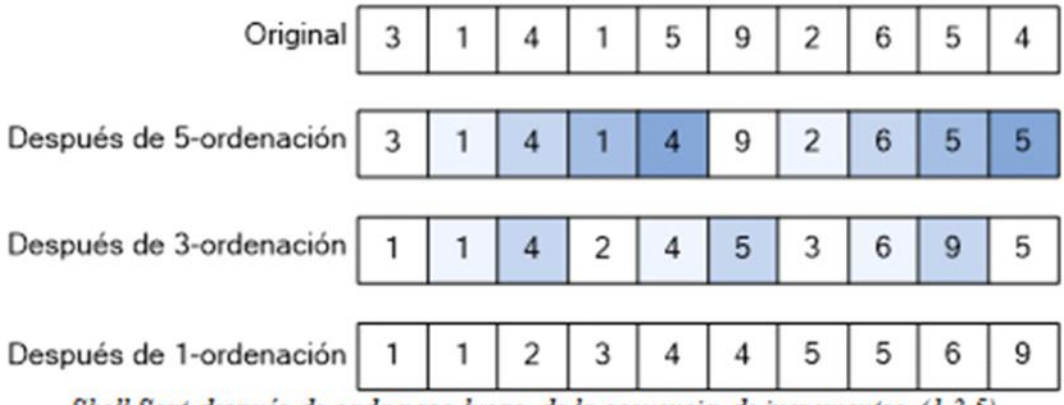
- Es una modificación del Insertion Sort que disminuye la cantidad de intercambios de elementos. Por ejemplo si hay un elemento muy pequeño muy a la derecha, justo en el lugar donde tendrían que estar los elementos más grandes, para moverlo hacia izquierda se necesitaría hacer cerca de n copias para llegar a la posición indicada.

- No todos los ítems deben ser movidos n espacios, pero en promedio deben moverse n/2 lugares. Por lo tanto lleva n veces n/2 cambios de lugar, resultando en n2/2 copias. Por lo cual el **tiempo de ejecución es O(n2)**.

- Para evitar la gran cantidad de movimientos, primero compara los elementos más lejanos y luego va comparando elementos más cercanos para finalmente realizar un Insertion Sort.

- Para lograr esto se utiliza una secuencia $H_1, H_2, \ldots\ H_n$ denominada secuencia de incrementos. Es importante remarcar que cualquier secuencia es válida siempre que H1=1, es decir que termine realizando un ordenamiento por inserción.

- Si bien no hay un consenso sobre la eficiencia del Shell Sort, **se considera que este varía entre O(n3/2) y O(n7/6)**



Shell Sort después de cada paso luego de la secuencia de incrementos (1,3,5).

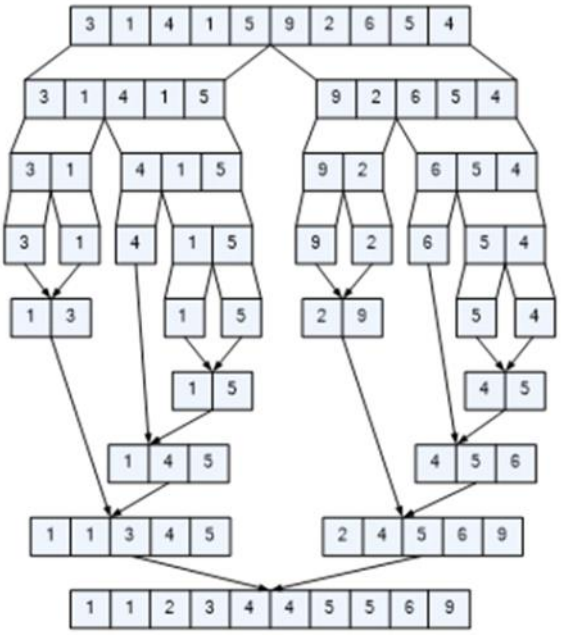
Merge Sort

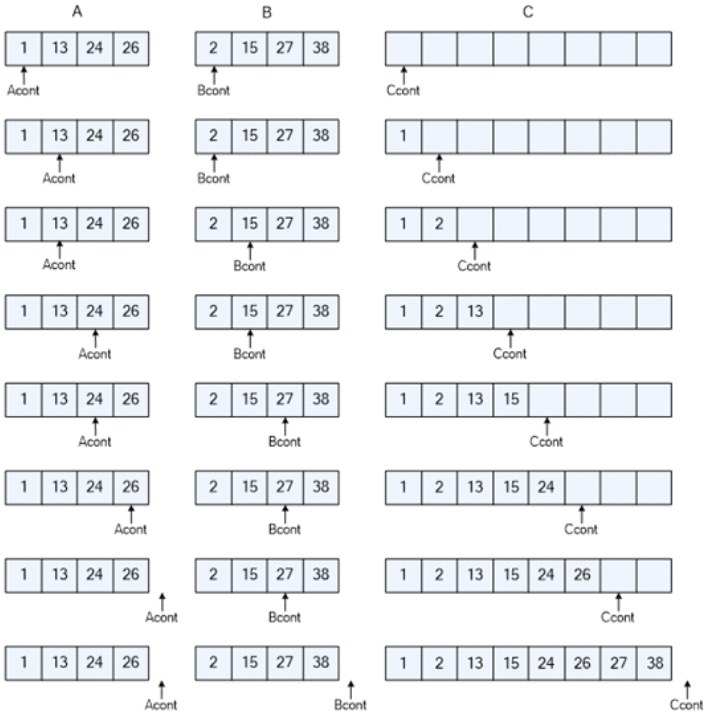
lunes, 28 de junio de 2021 09:12 p. m.

- Es un algoritmo recursivo que utiliza la técnica de divide y vencerás para obtener un **tiempo de ejecución O(n . logn)** sin importar cual sea la entrada.

- Se basa en la fusión de dos o más secuencias ordenadas en una única secuencia ordenada. Una de las desventajas de este algoritmo es que requiere de memoria extra proporcional a la cantidad de elementos del array. Es un algoritmo a considerar si estamos buscando velocidad, estabilidad, donde no se tolera un 'peor de los casos' y además disponemos de memoria extra. Algo que hace más atractivo a Merge Sort es que suele acceder de forma secuencial a los elementos y es de gran utilidad para ordenar en ambientes donde solo se dispone de acceso secuencia a los registros.

- El algoritmo tiene como caso base una secuencia con exactamente un elemento en ella. Y ya que esa secuencia esta ordenada, no hay nada que hacer. Por lo tanto para ordenar una secuencia n > 1 elementos se deben seguir los siguientes pasos:
    - Dividir la secuencia en dos subsecuencias más pequeñas.
    - Ordenar recursivamente las dos subsecuencias
    - Fusionar las subsecuencias ordenadas para obtener el resultado final.


3 1 4 1 5 9 2 6 5 4
3 1 4 1 5
9 2 6 5 4
3 1
4 1 5
9 2
6 5 4
3
1
4
1 5
9
2
6
5 4
1 3
1
5
2 9
5
4
1 5
4 5
1 4 5
4 5 6
1 1 3 4 5
2 4 5 6 9
1 1 2 3 4 4 5 5 6 9


A
B
C
1 13 24 26
2 15 27 38
Acont
Bcont
Ccont
1
1 2
1 2 13
1 2 13 15
1 2 13 15 24
1 2 13 15 24 26
1 2 13 15 24 26 27 38

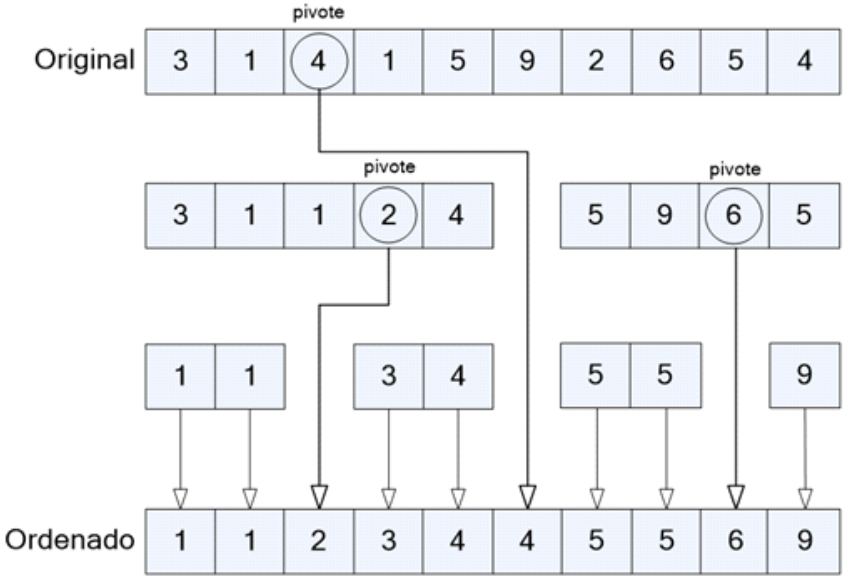
Quick Sort

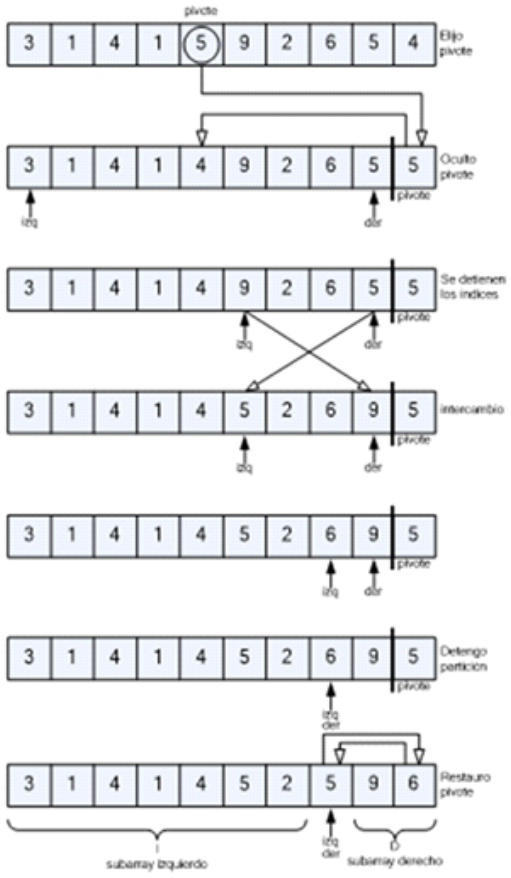
lunes, 28 de junio de 2021 09:19 p. m.

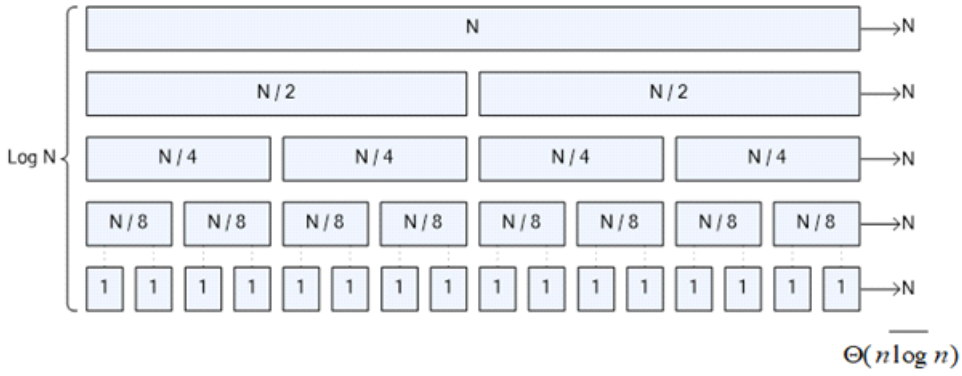
- Es el algoritmo que mejor responde en la mayoría de los casos, con un **tiempo promedio de O(nlogn) y O(n2) en el peor de los casos**.

- El QuickSort está basado en la idea de divide y vencerás, en el cual un problema se soluciona dividiéndolo en dos o más subproblemas, resolviendo recursivamente cada uno de ellos para luego juntar sus soluciones para obtener la solución del original.

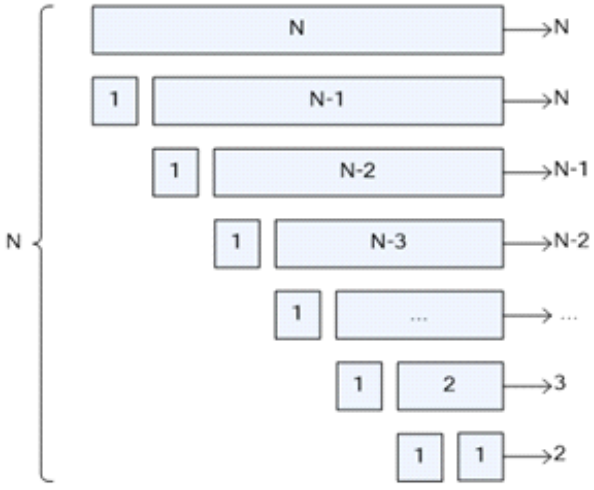
- El algoritmo básico consiste en los siguientes cuatro pasos:
  - Elegir el un elemento como pivote
  - Comparar todos los elementos con el pivote generando dos subconjuntos a izquierda los menores o iguales y a derecha los mayores que el pivote







$\Theta(n \log n)$



$\Theta(n^2)$

Bsort - Mean Sort

lunes, 28 de junio de 2021 09:21 p. m.

BSORT

- Es una variante del Quicksort donde el funcionamiento del método es igual pero solo cambia la elección del pivote que este caso es el elemento central.

MEANSORT

- Es una variante del Quicksort donde el funcionamiento del método es igual pero solo cambia la elección del pivote que este caso es el elemento más próximo a la media.

Heap Sort

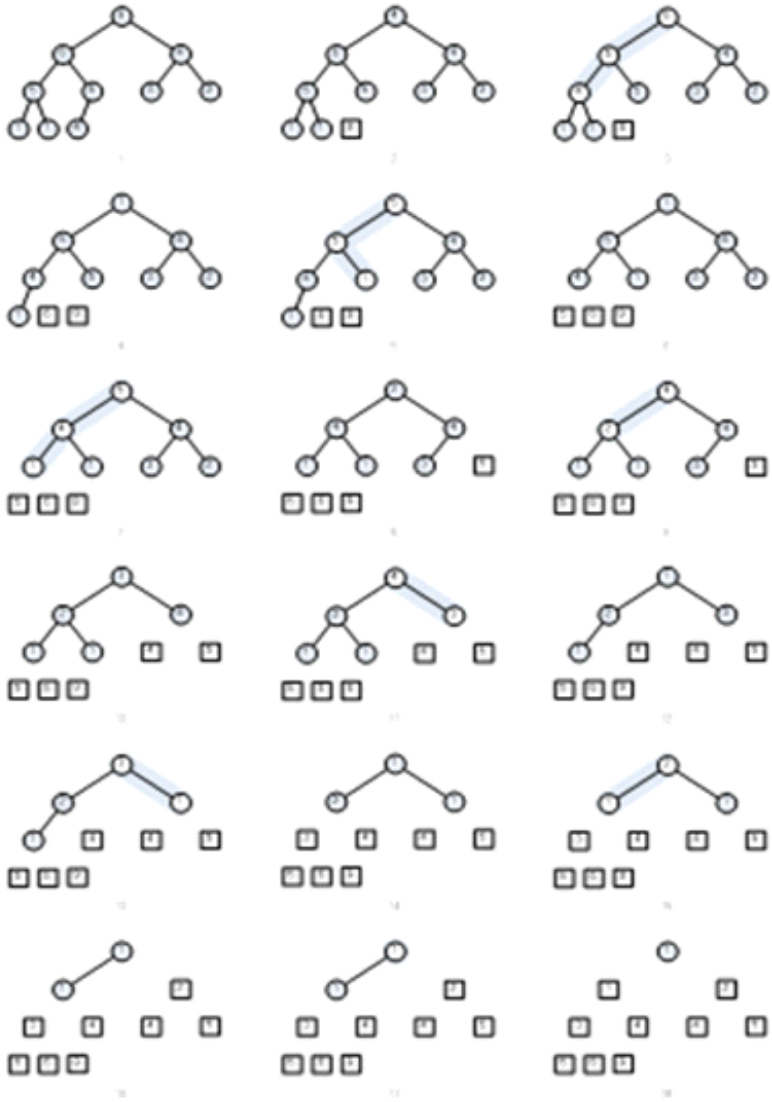
lunes, 28 de junio de 2021 09:26 p. m.

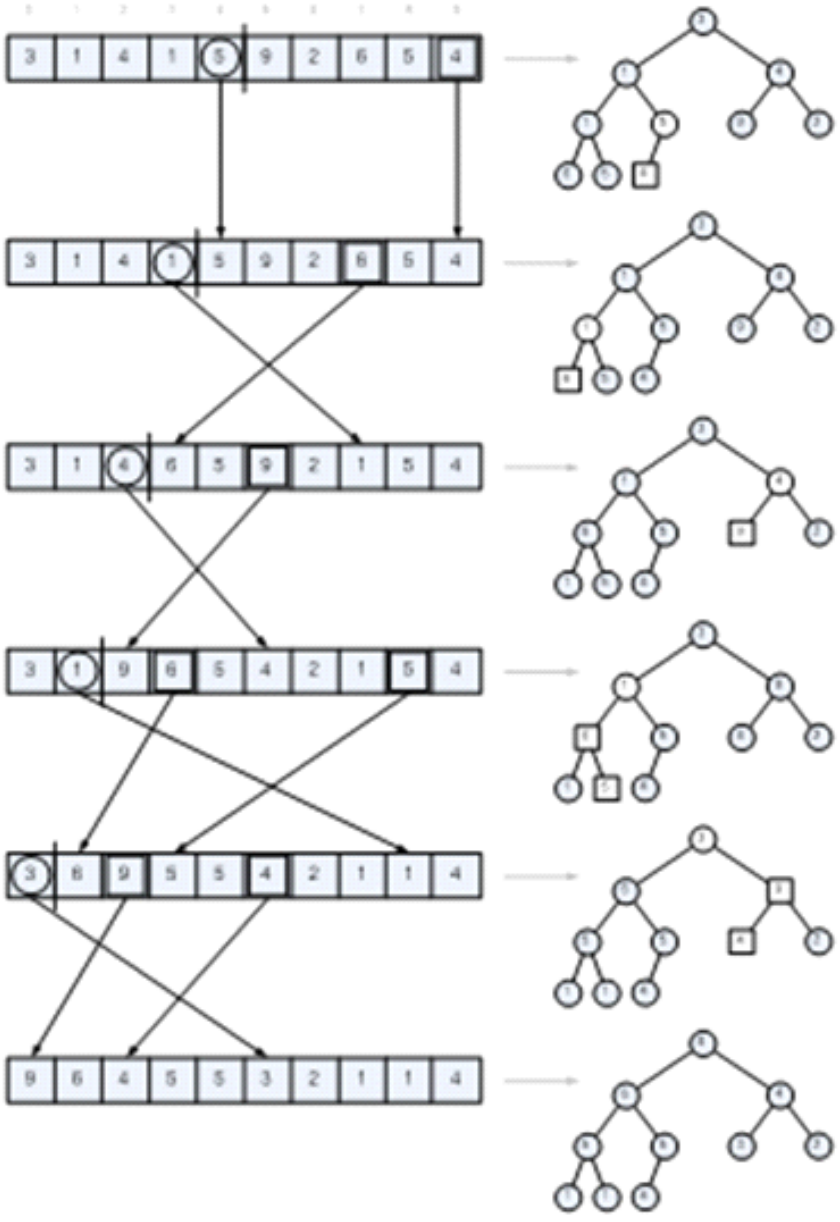
- Heap Sort, o clasificación por montículo (heap), se basa en una estructura de datos llamada montículo binario (heap), que es una de las formas de implementar una cola de prioridad.

- Un heap es un árbol binario completo en el cual la clave de cada nodo debe ser mayor (o igual) a las claves de sus hijos, si es que tiene. Esto implica que la clave más grande está en la raíz.

- **Este algoritmo tiene un orden de complejidad que nunca supera O(nlogn) y no requiere espacio de memoria adicional (in situ)**. Es generalmente se usa en sistemas embebidos con restricciones de tiempo real, o en sistemas en donde la seguridad es un factor importante.

- El algoritmo de Heap Sort consiste de dos fases.
  - En la primera fase, con los elementos a ordenar se construye un heap.
  - En la segunda fase, una vez construido el heap, se desarma dicho heap y de esa forma obtendremos los valores ordenados.




3 1 4 1 5 9 2 6 5 4
3 1 4 1 5 9 2 6 5 4
3 1 4 6 5 9 2 1 5 4
3 1 9 6 5 4 2 1 5 4
3 6 9 5 5 4 2 1 1 4
9 6 4 5 5 3 2 1 1 4

Unidad 5 - Índices

Concepto - Objetivo

martes, 29 de junio de 2021 10:37 a. m.

En las estructuras de datos se guardan los datos secuencialmente (uno atrás de otro), para no tener que buscar de manera secuencial un dato se crean índices.

Objetivo

- **Acceder a un dato determinado de forma directa**, **evitando una búsqueda secuencial** (esto es muy ineficiente ya que se debe buscar comparando uno a uno los datos almacenados)

- El objetivo de los índices **NO** es guardar los datos de manera ordenada en la estructura de datos, como así tampoco listar de manera ordenada los datos que se encuentran almacenados.

- El objetivo es crear una estructura adicional a la tabla que permita mantenerlos los datos ordenados en función de alguna clave.

    - Por ejemplo la creación de una PRIMARY KEY o un índice UNIQUE

- El objetivo de un índice es físico no lógico, esto quiere decir que busca físicamente acceder a la información de forma más rápida.

    - **NO** es un índice lógico que pretende tener ordenados lógicamente los elementos y así poder verlos.

    - Es un **componente físico** que nos permite acceder más rápido a la información.

Concepto

- Dependiendo la arquitectura de hardware utilizada, el índice puede formar parte de la tabla o ser una estructura adicional a la tabla sobre la cual se crea, lo que genera un espacio adicional y la necesidad de incorporar y eliminar valores en ambos sitios.

- Para el caso de la Arquitectura de PC (8086) el índice es una estructura adicional a la tabla.

Almacenamiento

martes, 29 de junio de 2021 11:38 a. m.

Cuando se ingresan filas en una tabla por ejemplo estas filas:

- 14 , CARLOS, ….
- 2 , PEDRO, …..
- 5 , JOSE, ….
- 24 , MARIA, …..
- 56 , LUCIA, …..
- 3 , MIRTA ,…..

Pos	Id	Nombre	Resto
0	14	CARLOS	…….
1	2	PEDRO	…….
2	5	JOSE	…….
3	24	MARIA	…….
4	56	LUCIA	…….
5	3	MIRTA	…….

- Podemos ver como **se almacena la información en el orden secuencial** en el que fueron ingresados.

- *El campo "POS" representa la posición relativa del registro en la tabla, se dice que es relativa porque por ejemplo el registro POS = 5 NOMBRE = "MIRTA" ocupa la quinta posición con respecto al inicio, se encuentra almacenado a 5 posiciones respecto de la posición de inicio.*

- *En la computación todo el almacenamiento se considera relativo porque todo se expresa de manera relativa a un inicio. No se expresa el valor absoluto de la posición donde se almacena la información.*

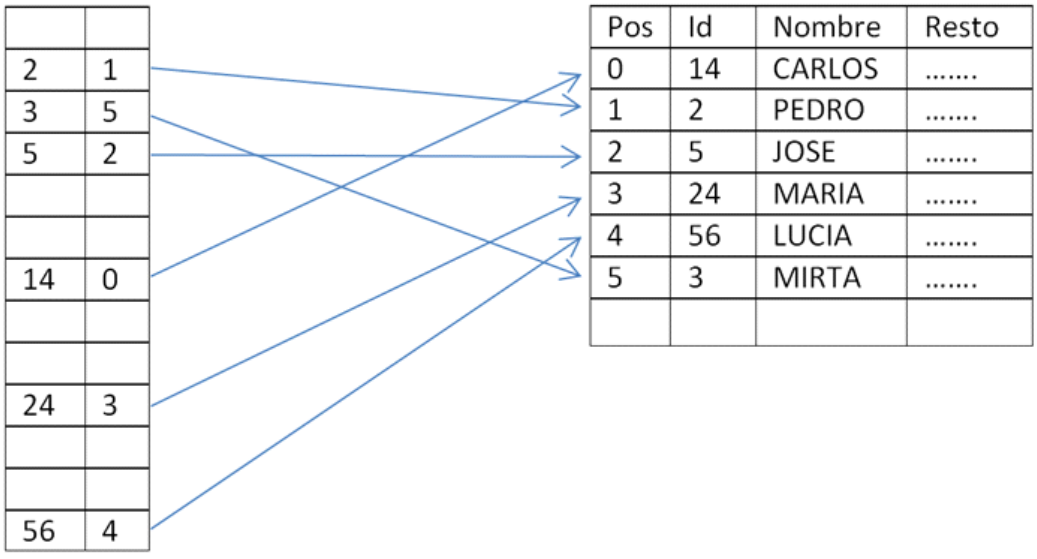
    - *Por ejemplo: la posición [3] de un vector expresa que se encuentra a 3 posiciones respecto del inicio.*

Creando el índice

- Al crear un **índice** se **genera** una **nueva estructura** la que nos **permite acceder a los datos de manera directa**.

- En esta nueva estructura **los datos se ingresaron de manera ordenada según el campo "ID"** (primer columna de la tabla índice) y por otro lado también **se almacena** una **referencia** a la **posición relativa** de los **registros** en la **tabla original** donde están los datos (segunda columna de la tabla índice).

- Para no hacer una duplicidad de los datos en la tabla de índice solo se almacena la posición relativa.



2	1
3	5
5	2
14	0
24	3
56	4

Pos	Id	Nombre	Resto
0	14	CARLOS	.....
1	2	PEDRO	.....
2	5	JOSE	.....
3	24	MARIA	.....
4	56	LUCIA	.....
5	3	MIRTA	.....

Tipos de Acceso

martes, 29 de junio de 2021 12:04 p. m.

Tipos de Acceso: existen diferentes formas de **acceder** a los **datos** en la computación.

- **Secuencial**
  - El **acceso** se realiza **en función al modo en que ingresaron los datos**.
  - Típico FOR - IF que recorre completamente estructuras secuencialmente.

- **Secuencial Indexado**
  - Se aplica una **búsqueda secuencial** pero **recorriendo un índice**.
  - Permite devolver los datos en orden por el índice.
    - El acceso se realiza en función de alguna clave que fue definida (clave del índice).

- **Directo o Random**
  - El **acceso** es en **forma directa** a una clave sin realizar ningún recorrido.

Métodos - Creación de índices

martes, 29 de junio de 2021 12:31 p. m.

En la práctica existen **dos métodos para crear índices en un DBMS** (Data Base Managment System).

- **Hashing (tablas de hashing)**
    - En la arquitectura de **Mainframe** y **Minicomputadoras** (estructuras **estáticas**) el método utilizado es **Hashing**.
    - Las estructuras **estáticas** poseen los **recursos asignados previamente al inicio**, ya que **tienen un fin o funcionalidad definidos y específicos**.
        - Esto les permite manejar los recursos de manera más eficiente que las estructuras dinámicas.
- **Árbol B (Btree)**
    - En la **arquitectura 8086** utilizada en las **PC** (estructuras **dinámicas**) el método utilizado es el de **Árbol B**.
    - Las estructuras **dinámicas asignan recursos a medida que sean necesarios** ("On-Demand") ya que estas computadoras son **utilizadas para múltiples usos y funcionalidades distintos**, por lo tanto manejan los recursos de manera más ineficiente que las estructuras estáticas.

Hashing

martes, 29 de junio de 2021 12:35 p. m.

- El método de **Hashing** trabaja sobre el **concepto de una tabla y una función hash**, también llamada función de dispersión, dicha función se utiliza para **convertir algún tipo de dato en un pequeño número que puede servir como huella digital de ese dato**.

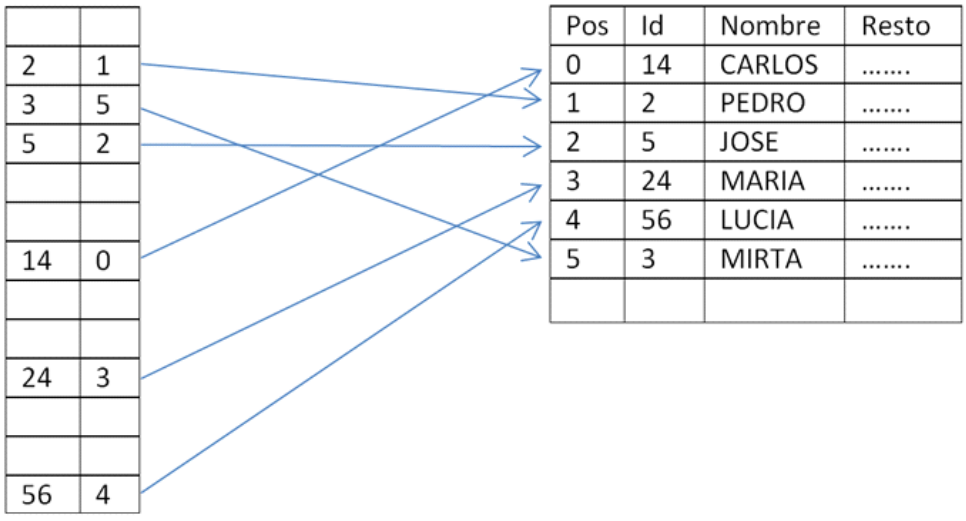
- El **término hash** proviene, aparentemente, de la analogía con el significado en inglés de dicha palabra en el mundo real: picar y mezclar. El **algoritmo de hash "pica y mezcla" los datos para crear las huellas digitales**.

    - Estas son llamadas **valores o códigos hash**, los cuales **pueden ser utilizados como índices en tablas hash** o bien como controles de integridad de datos o archivos.

Método de Hashing

- El método crea una tabla (vector de dos dimensiones) donde en la primer dimensión colocará las claves y en la segunda dimensión las posiciones relativas de las mismas en la tabla donde se encuentran los datos correspondientes a esa clave.

El método crea una estructura adicional donde mantiene la clave ordenada y la posición relativa a los datos:



2	1
3	5
5	2
14	0
24	3
56	4

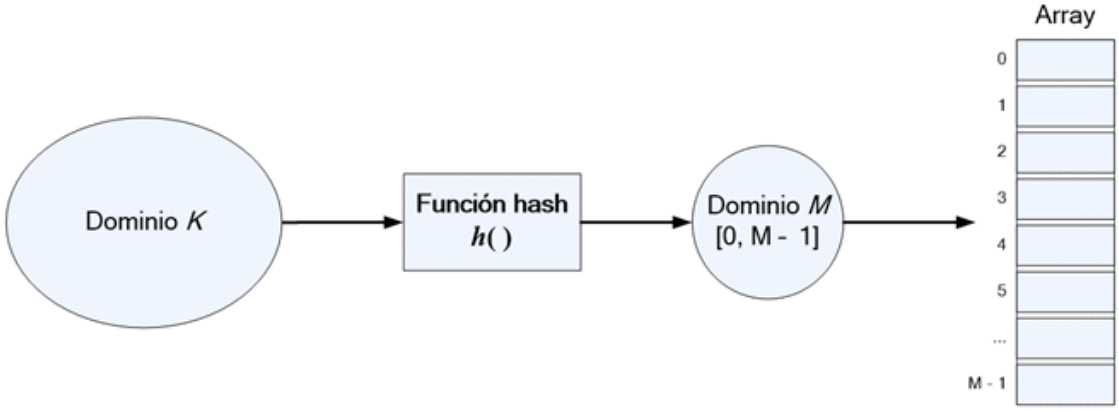
Pos	Id	Nombre	Resto
0	14	CARLOS	.....
1	2	PEDRO	.....
2	5	JOSE	.....
3	24	MARIA	.....
4	56	LUCIA	.....
5	3	MIRTA	.....

Función de hash()

martes, 29 de junio de 2021 01:24 p. m.

- La **función hash()** recibe como **entrada la clave a almacenar** y **devuelve un valor numérico entero** que **corresponde a la posición en la cual debería ir dicha clave** en la tabla mencionada.

- El valor numérico entero calculado debe ser único para cada valor del dominio de entrada y no cambiará a lo largo del tiempo.



Se considera que una buena función de hash es aquella que tiene las siguientes cualidades:

- **Evitar colisiones**
    - Se cumple cuando **dado un conjunto de valores de entrada**, los resultados obtenidos en el **conjunto de salida son distintos**. Nunca puede devolver el mismo valor de salida para distintos valores de entrada.
    - La función debe ser biyectiva (inyectiva y sobreyectiva) o inversible (Codominio y Dominio de mismo tamaño), para que no devuelva el mismo valor de salida para distintos valores de entrada.
- **Tiende a distribuir las claves uniformemente**
    - Indica que tiende a distribuir uniformemente los valores de salida respecto a los valores de entrada.
- **Es fácil de calcular**
    - No necesariamente significa que sea fácil de escribir el algoritmo para calcular la función, sino que significa que el tiempo de ejecución de la función de hash debe ser O(1).

Colisiones de Hashing

martes, 29 de junio de 2021 12:50 p. m.

- Si ocurriese que una clave generara un valor de hash que apuntase a una posición de la tabla ya ocupada, estaríamos en presencia de una **colisión**.

- Esto suele ocurrir si el Codominio (tabla de hash) es más reducido que el tamaño del Dominio (tabla de datos original), entonces de esta manera la función de hash generara mismos valores de salida para distintos valores de entrada.

    - Por ejemplo: Si se almacenan distintos CHAR de 30 caracteres (256^{30} combinaciones distintas de caracteres), cuando la función de hash quiera asignar un numero natural a una determinada entrada es muy probable que valor que devuelva sea el mismo que para otro elemento del Dominio. Ya que el Codominio (espacios en la tabla de hash o índice) es mucho más acotado que el Dominio de entrada.

- En este caso no podríamos almacenar dos registros en una misma posición, por lo que debemos encontrar otra ubicación en donde almacenar ese nuevo registro a través de las técnicas de resolución de colisiones.

Hay varias **técnicas de resolución de colisiones**, entre ellas se encuentran:

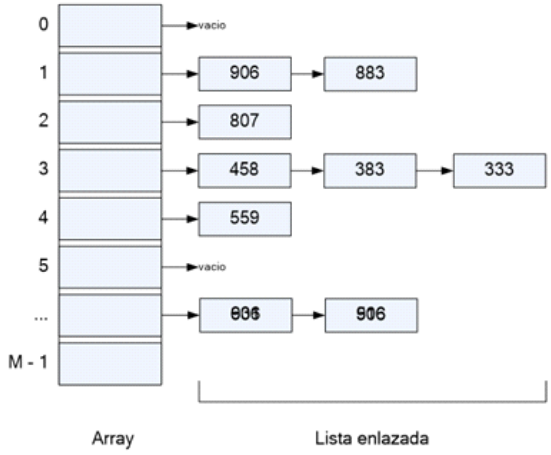
- **Encadenamiento**
- **Direccionamiento abierto**

Encadenamiento

Es la **técnica de resolución de colisiones más simple**.

Cada casilla en el vector referencia una lista de los registros insertados que colisionan en la misma casilla.

- La inserción consiste en encontrar la casilla correcta (con la función hash) y agregar al final de la lista correspondiente.
- La consulta consiste en encontrar la casilla donde se almacenaron los datos (con la función hash) y leer el registro en la tabla original con el puntero a su posición relativa.
- El borrado consiste en buscar (con la función hash) y quitar de la lista.



- **Desventaja**
    - Ocupa más espacio del necesario y ya reservado con la tabla de hash predimensionada.
- **Ventajas**
    - El acceso es mucho más rápido que los sondeos del direccionamiento abierto.
    - Se puede respetar el concepto de secuencialidad indexada, las claves se encontraran agrupadas en un orden lógico.
        - Por ejemplo se podrían si se almacenaran números y se quisieran obtener todos los números mayores a 1000, entonces se deberá calcular la posición del valor 1000 a través de la función de hash y a partir de esa posición se recorrerá secuencialmente el vector de hash.

Direccionamiento abierto

En el direccionamiento abierto, **cuando un registro no puede ser ubicado en el índice calculado por la función de hash, se busca otra posición dentro del mismo vector o tabla de hash**. No utiliza estructura adicionales de almacenamiento.

Hay varios **métodos de elección de esa posición**, los cuales varían en el método para buscar la próxima posición. Los más usados son:

- **Sondeo lineal**
- **Sondeo cuadrático**
- **Hashing doble**

- **Desventajas**
    - No respeta el concepto de secuencialidad indexada, es probable que las claves no se encuentren agrupadas en un orden lógico debido a los saltos realizados por los sondeos o el hashing doble.

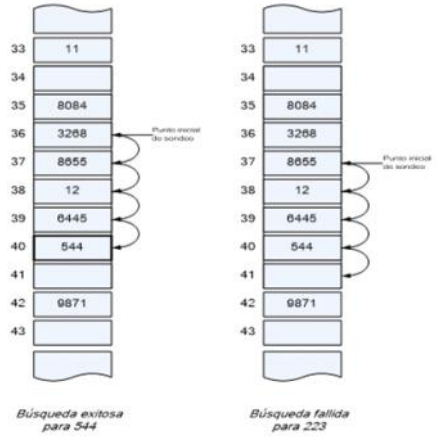
- **Ventajas**
    - No ocupa más espacio que el espacio reservado previamente al crear la tabla de hash.

- **Sondeo lineal**

La **forma más simple de resolver el problema** es el **sondeo lineal**, el cual **busca secuencialmente** en la tabla hasta encontrar **una posición vacía**.

En caso de alcanzar la última posición de la tabla esta continúa buscando desde la primera posición de la tabla.

Este tipo de sondeo no permite una buena distribución de las claves colisionadas, esto significa que los valores se encontrarán más condensados o acumulados en algunos puntos de la tabla de hash.



- **Sondeo cuadrático**

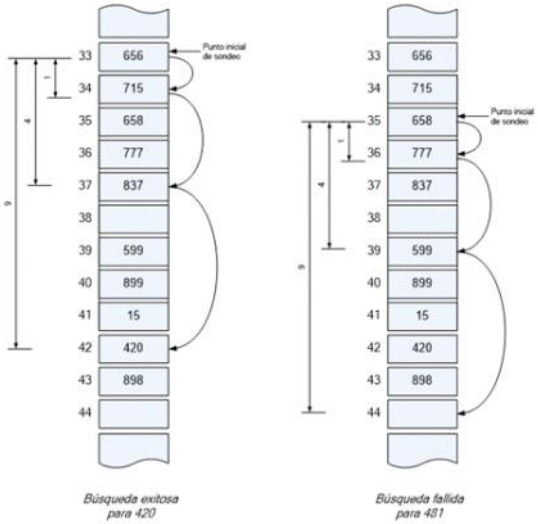
Ubica el valor que generó la colisión buscando una nueva posición a una distancia específica del punto inicial de sondeo (valor que calculó la función de hash).

Esta variación permite una mejor distribución de las claves colisionadas, esto significa que los valores se encontrarán más distribuidos a lo largo de la tabla de hash.

El nombre cuadrático deriva de la fórmula utilizada para resolver las colisiones: $F(i) = i^2$

$F(i)$ representa el desvío cuadratico utilizado en cada evaluación del sondeo y H representa el valor calculado por la funcion de hash.

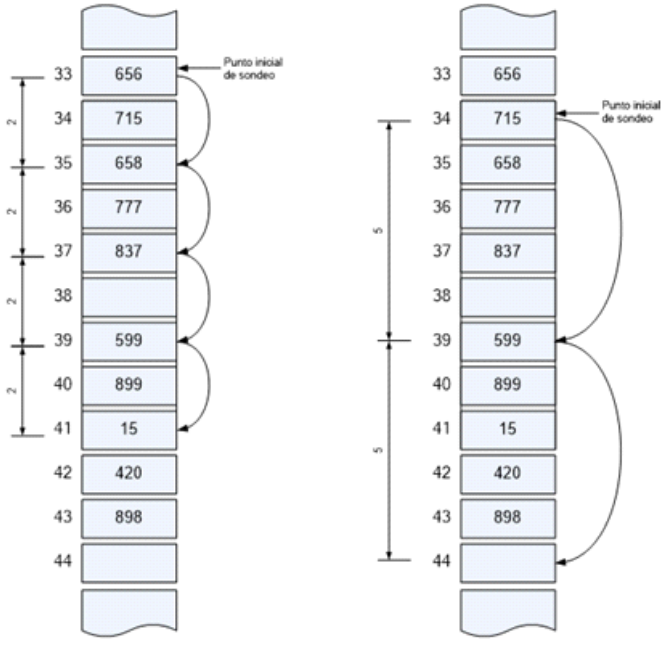
Si resulta que la función de hash evalúa a la posición y la posición es inconclusa,
se intentan las posiciones $H + 1^2, H + 2^2, H + 3^2, \dots, H + i^2$



- **Hashing doble**

Aplica una función hash a la clave una segunda vez, usando una **función de hash distinta** y usa ese resultado como **tamaño de salto**.

Para que este método arroje buenos resultados **la función de hash secundaria debe ser distinta que la primaria y debe arrojar valores mayores a cero** (sino no habría saltos, y se produciría un bucle infinito).



Arboles M-arios

martes, 29 de junio de 2021 02:31 p. m.

- Los árboles M-arios son árboles que tienen grado mayor a 2.

- Si un árbol tiene un grado mayor que 2 puede almacenar en menos niveles la misma información que un árbol binario, entonces la búsqueda de un dato dentro del árbol M-ario será mucho más rápida en comparación a la misma búsqueda en un árbol binario.

    - Para que la búsqueda sea lo más rápida y eficiente posible en un árbol M-ario este debe estar lo más completo y balanceado posible.

- *Si se tiene un conjunto de datos muy grande, tan grande que no podemos colocarlo en memoria principal, nos veríamos obligados a implementar el árbol de búsqueda en un almacenamiento secundario, como el disco.*

    - *Las características de un disco a diferencia de la memoria principal hacen que sea necesario utilizar valores de M más grandes para poder implementar estos árboles de búsqueda de forma eficiente.*

- *El tiempo de acceso de un disco típico es de 1 a 10 ms, mientras que el tiempo de acceso típico de una memoria principal es de 10 a 100 nano segundos. Por lo tanto, los accesos a memoria son entre 10.000 y 1.000.000 de veces más veloces que los accesos a disco. Para maximizar la performance es necesario minimizar a toda costa los accesos a disco.*

- *Además, los discos son dispositivos orientados a bloques. Los datos son transferidos en bloques de gran tamaño entre la memoria principal y el disco. Los tamaños de bloques típicos varían entre 512 y 4096 bytes. En consecuencia, tiene sentido tomar ventaja de esa habilidad para transferir grandes bloques de datos eficientemente.*

Árboles B

martes, 29 de junio de 2021 02:33 p. m.

- El **árbol B** es un tipo de **árbol M-ario** destinado a la **creación de índices físicos** para el acceso a la información.

- El **objetivo principal** es **minimizar** las **operaciones** de entrada y salida **hacia el disco**.

- Al imponer la **condición de balance**, el **árbol es restringido** de manera tal que se garantice que la **búsqueda**, la **inserción** y la **eliminación** sean de tipo **logarítmica** y de **tiempo:** $\boldsymbol{O(log\ n)}$

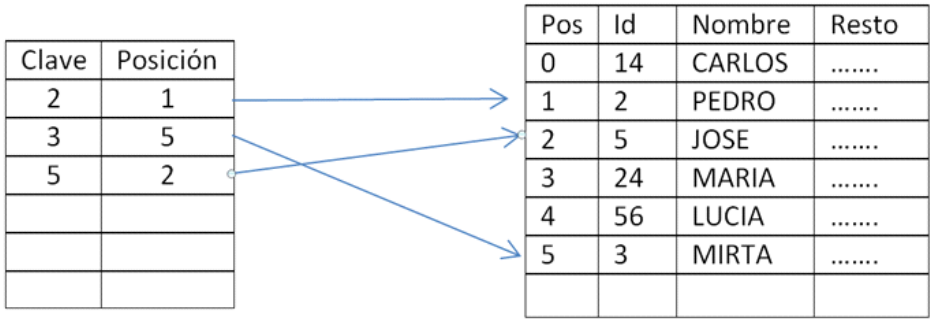
- El **grado M** del árbol B varía entre **50** y **2000** y **se determina en base al tamaño de las claves** y del **tamaño de la página del disco**.

Nodos

El **árbol B** es uno de los pocos árboles con **dos tipos de nodos diferentes**:

- **Nodo Hoja**

Tiene una componente de dato donde van los **valores de las claves** y un componente **puntero** que contiene la **posición relativa de los datos secuenciales correspondientes a esa clave**. Los registros se insertan **ordenados de menor a mayor** según el valor de la clave.

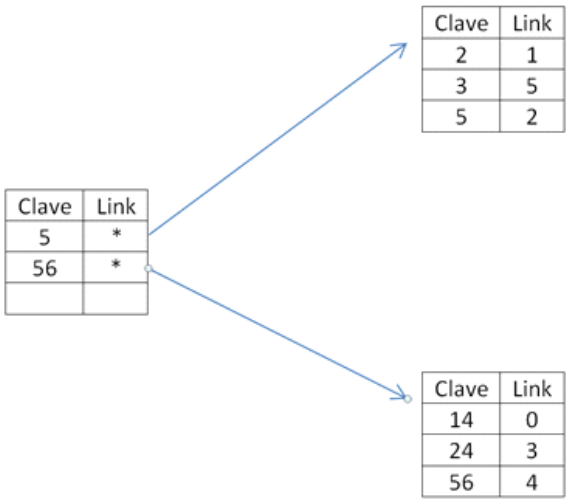


Clave	Posición
2	1
3	5
5	2

Pos	Id	Nombre	Resto
0	14	CARLOS	.....
1	2	PEDRO	.....
2	5	JOSE	.....
3	24	MARIA	.....
4	56	LUCIA	.....
5	3	MIRTA	.....

- **Nodo Raíz o Rama**

Tiene una componente de dato donde van los **valores de las claves ordenados de menor a mayor** y un componente **puntero** que **apunta al nodo que contiene claves menores o iguales que ella**.



Un **árbol-B comienza** su crecimiento con un **nodo Hoja** a diferencia de los demás tipos de árboles que comienzan su crecimiento desde un **nodo Raíz**.

Ya que **al comenzar el árbol-B solo se almacenan claves de la tabla original y sus posiciones relativas**, entonces se necesita un **nodo Hoja** para almacenar esta información.

El **primer nodo del árbol-B** es **realmente** (siempre) un **nodo Hoja**, pero a la vez es un **nodo Raíz circunstancialmente** (por el momento) hasta que ingresen nuevos valores y este pase a ser solo un **nodo Hoja**; generándose un nuevo **nodo Raíz** el cual tendrá un puntero al **nodo Hoja.** El **árbol-B** crece desde las hojas hacia la raíz.

Distinto es el caso de los otros tipos de árboles, donde el **primer nodo del** es **realmente** (siempre) un **nodo Raíz**, pero a la vez es un **nodo Hoja circunstancialmente** (por el momento) hasta que ingresen nuevos valores y este pase a ser solo un **nodo Raíz**; generándose un nuevo **nodo Hoja** el cual será apuntado por el **nodo Raíz.** Estos árboles **crecen** desde la **raíz** hacia las **hojas**.

Árbol B - Búsqueda

martes, 29 de junio de 2021 08:08 p. m.

Búsqueda

Buscar en un **árbol-B** es muy parecido a buscar en un árbol binario de búsqueda, excepto que en vez de hacer una decisión binaria, o de dos caminos en cada nodo, hacemos una decisión multi camino.

Cuando se quiere buscar una determinada clave se **comienza** por el **nodo Raíz** y se continua buscando a través del **puntero** que tenga **el primer valor de clave mayor o igual** que el buscado **sucesivamente a través de los nodos Ramas** hasta alcanzar el **nodo Hoja** en el que se encuentra almacenada la **clave** y por lo tanto su **puntero** a la **posición relativa** en la tabla original.

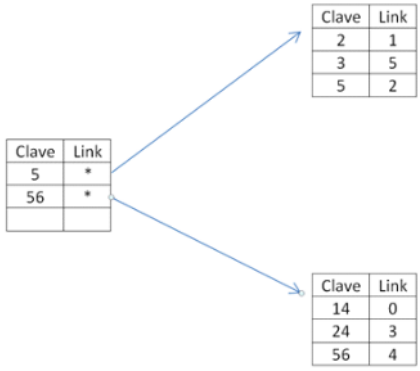
Este método es **más lento** que el método de búsqueda de hashing, pero si el **árbol-B** tiene **pocos niveles** el **tiempo se reduce bastante** aproximándose al tiempo que tardaría por hashing. Si el árbol-B tiene **muchos niveles** esto implica **acceder a memoria** por cada nodo de cada nivel y esto genera que **aumente el tiempo** de la búsqueda.

Por ejemplo:

Si se quiere buscar el registro de **Clave = 3** se comienza a buscar por el **nodo Raíz** buscando el **primer valor de Clave mayor o igual a 3**, en este caso es el primer registro del **nodo Raíz** con **Clave = 5**.

Luego se accede al nodo linkeado por el registro de **Clave = 5** y se busca el **primer valor de Clave mayor o igual a 3**. En este nodo se encuentra el registro de **Clave = 3**, entonces se accede al **registro en la tabla original** a través del **puntero** a su **posición relativa** en la tabla original.

En el caso en el que se llegara a un **nodo Hoja** y **no se encontrara el valor de Clave buscado** después de recorrer todos los registros del nodo se concluye que **no se insertó nunca un registro con ese valor de Clave** en la tabla original y por lo tanto tampoco en el árbol-B.



Árbol B - Inserción

martes, 29 de junio de 2021 08:09 p. m.

Inserción

- ***Inserción (sin necesidad de Split)***

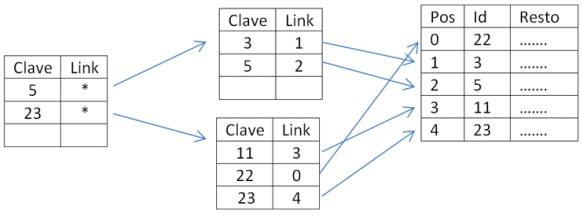
Para **insertar** un elemento comenzamos en el **nodo Raíz** y realizamos una búsqueda para él. Asumiendo que el elemento no está previamente en el árbol, la búsqueda sin éxito terminará en un **nodo hoja**. Este es el punto en el árbol donde el nuevo elemento será insertado.

- ***Split***

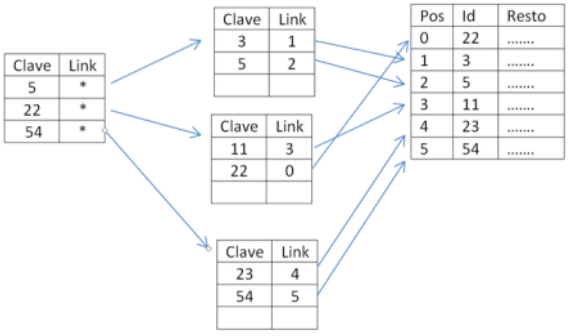
Si ocurre que cuando se llega al **nodo Hoja no hay espacio** para insertar el elemento se produce lo que se denomina **Split** que es un proceso que **divide el nodo en dos** dejando la **mitad de elementos en cada uno** respetando el **orden de menor a mayor**, quedando la **mitad de los elementos más chicos en un nodo** y la **mitad de los elementos más grandes en el otro**.

Por ejemplo:

Se inserta el elemento de **Clave = 23 sin necesidad de Split** y el árbol-B resulta de la siguiente manera:



En este caso podemos observar cómo actúa la **inserción** con **Split** al insertar el elemento de **Clave = 54**:



Árbol B - Eliminación

martes, 29 de junio de 2021 07:18 p. m.

Eliminación

- ***Eliminación***

Para **eliminar** un elemento comenzamos en el **nodo Raíz** y realizamos una búsqueda para él. Asumiendo que el elemento existe, si existe se llegará a la hoja donde está y se borra, sino se dirá que no existe.

- ***Fusión***

Si ocurre que cuando se elimina el elemento el **nodo Hoja** donde se encontraba queda vacío, debe eliminarse ese nodo, lo que puede generar una baja potencial en todos los antecesores de dicho nodo.

Árbol B - Caso Práctico

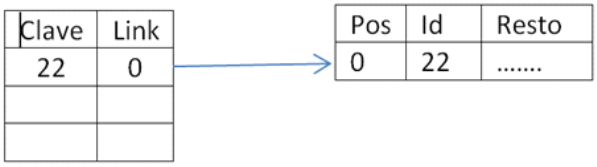
martes, 29 de junio de 2021 02:40 p. m.

Dadas los siguientes números de claves posibles realizaremos el proceso de inserción de elementos del árbol. A modo práctico lo realizaremos con un árbol de grado 3 para ver las diferentes situaciones que pueden ocurrir.

Orden de entrada de los datos:

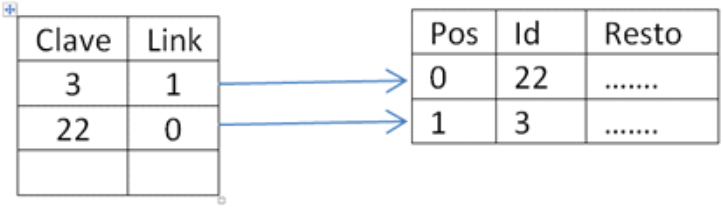
22 – 3 – 5 – 11 – 23 – 54 – 10 – 14 – 15 – 7 – 3 – 9

- Se inserta el elemento de **Clave = 22** en el **nodo Raíz**:

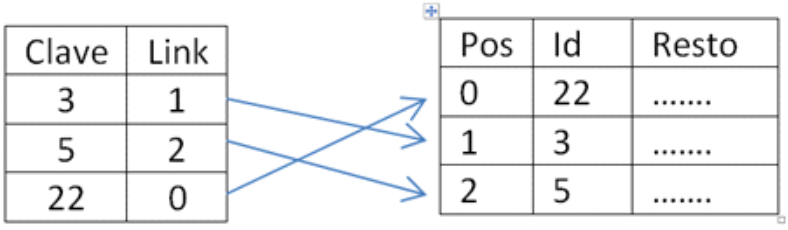


- Se inserta el elemento de **Clave = 3** antes que el de **Clave = 22** en el **nodo Raíz**:

(están mal las flechas azules de posición relativa solo en esta imagen por error de gráfico)

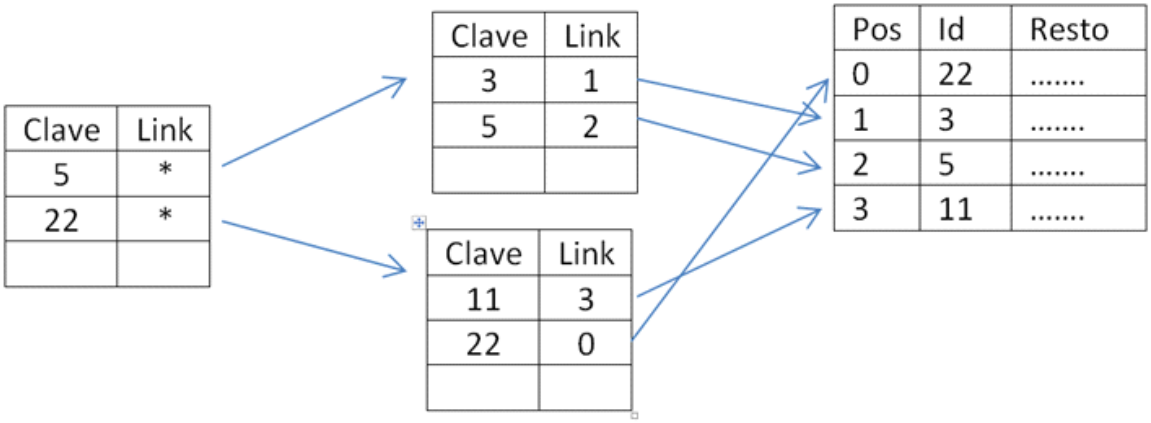


- Se inserta el elemento de **Clave = 5** antes que el de **Clave = 22** pero después del de **Clave = 3** en el **nodo Raíz**:

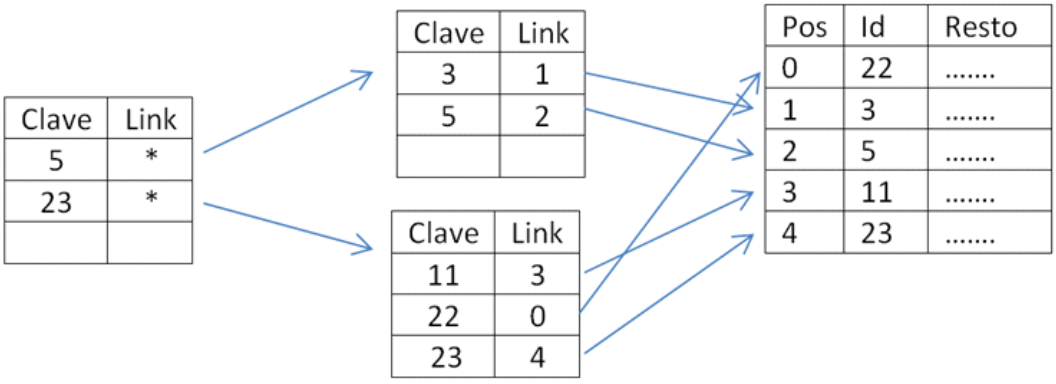


- Al insertar el elemento de **Clave = 11** se realiza un **Split**, ya que el **11** debería insertarse entre la **Clave 5 y 22.**

- Entonces se debe **dividir** el nodo en **dos nodos Hoja** y **crear un nuevo nodo Raíz** en el que estarán las **dos Claves de mayor valor existente en cada nodo Hoja** y un **puntero** a estos **nodos Hoja** resultantes del **Split** realizado**:**

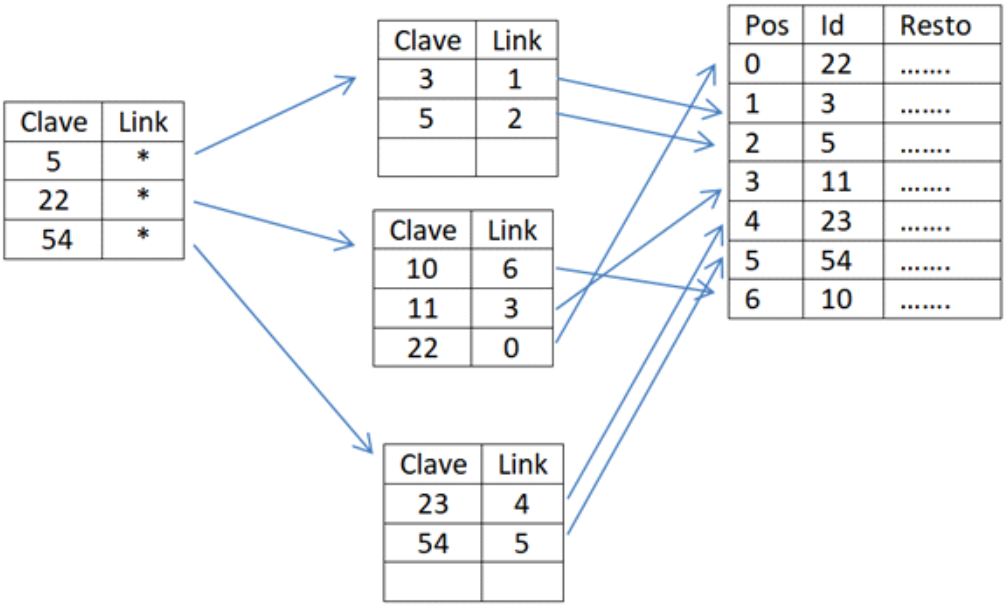


- Se inserta el elemento de **Clave = 23** después del elemento de **Clave = 22** en el **nodo Hoja**:

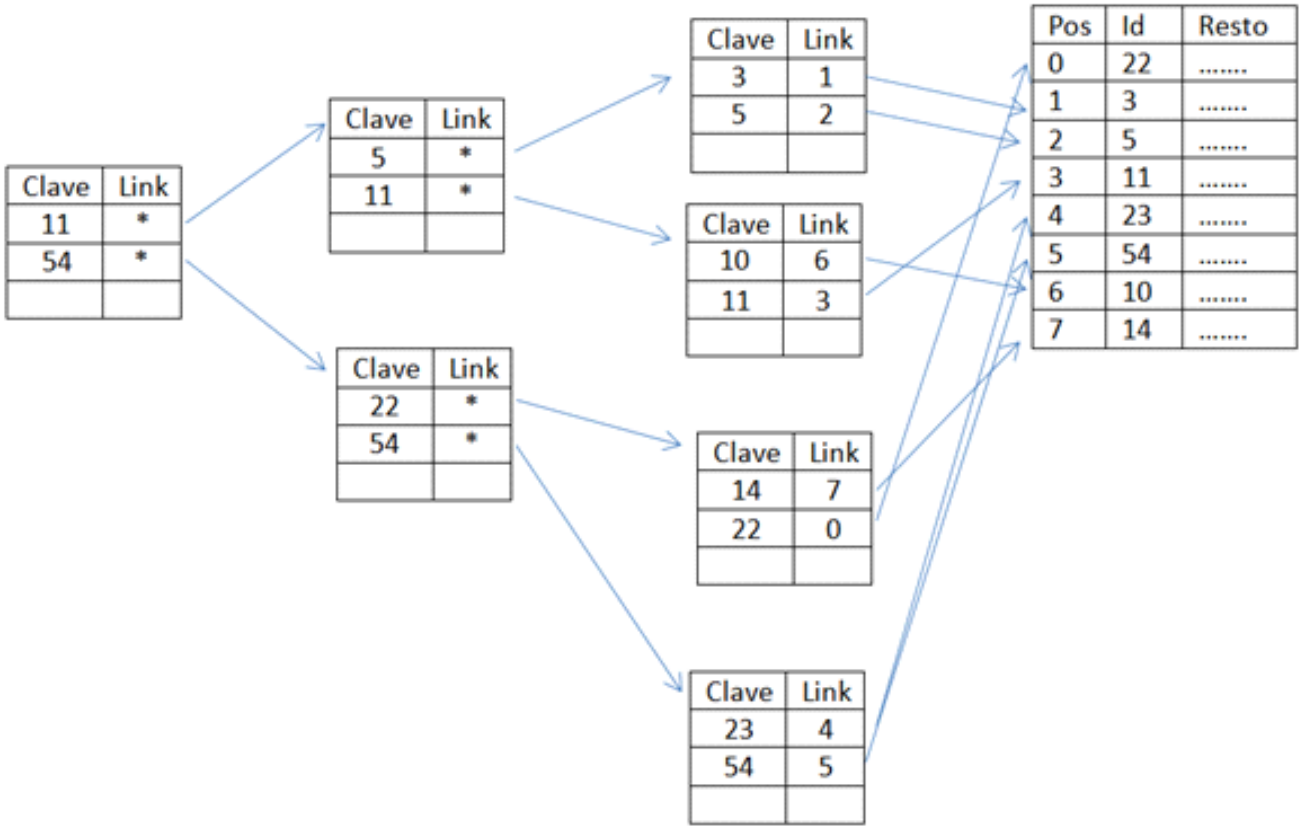


- Al insertar el elemento de **Clave = 54** se realiza un **Split**, ya que el **54** debería insertarse en el nodo que se encuentra la **Clave = 23.**

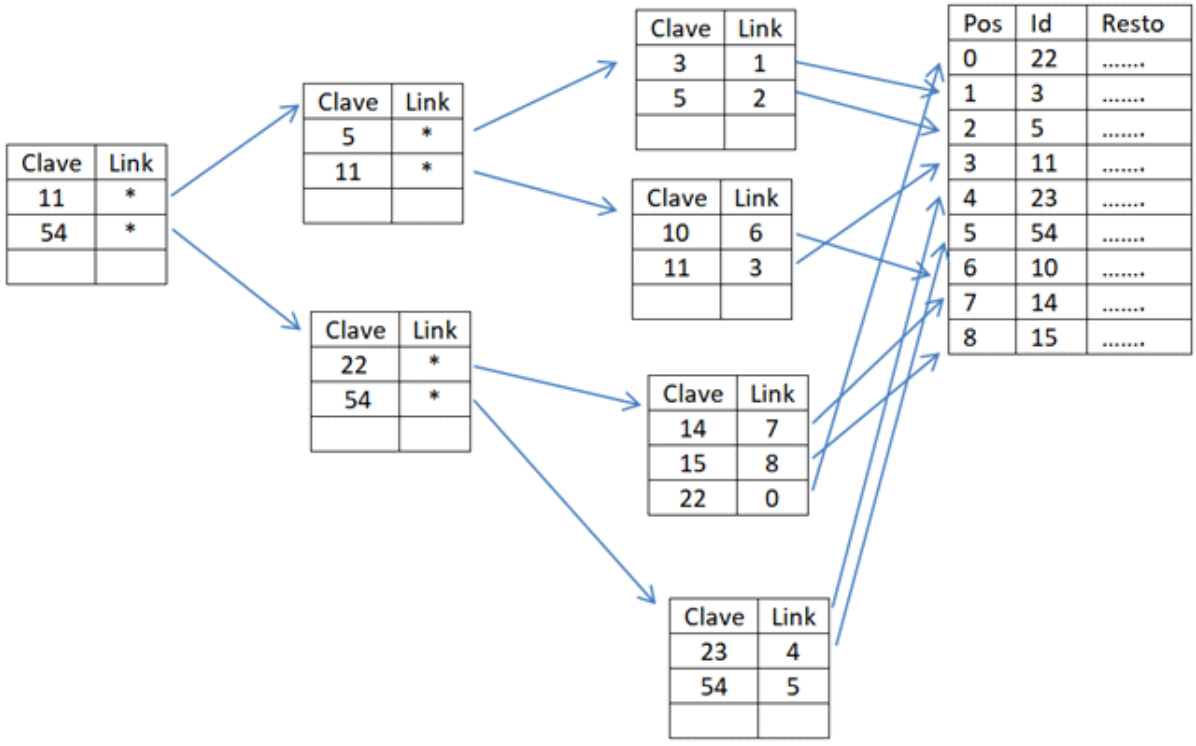
- Entonces se debe **dividir** el nodo en **dos nodos Hoja** de la siguiente manera**:**



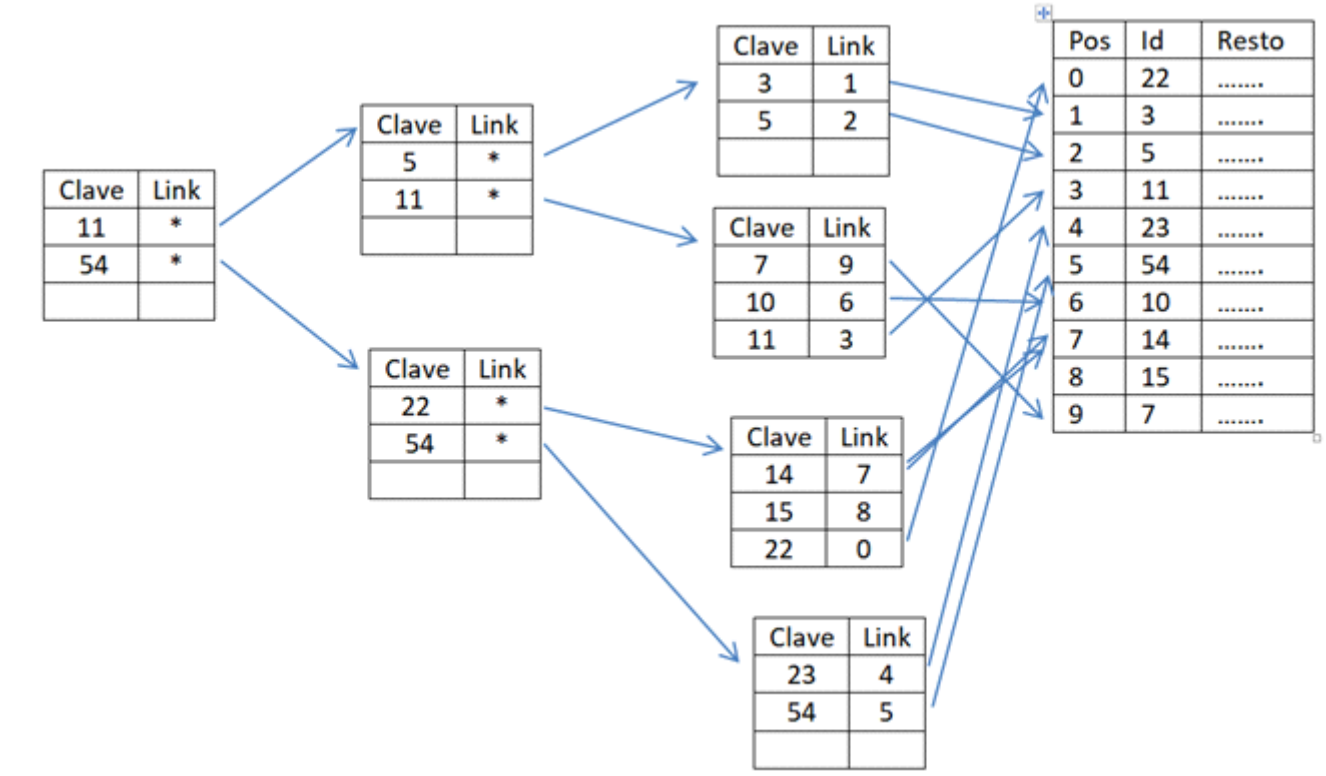
- Se inserta el elemento de **Clave = 14**:



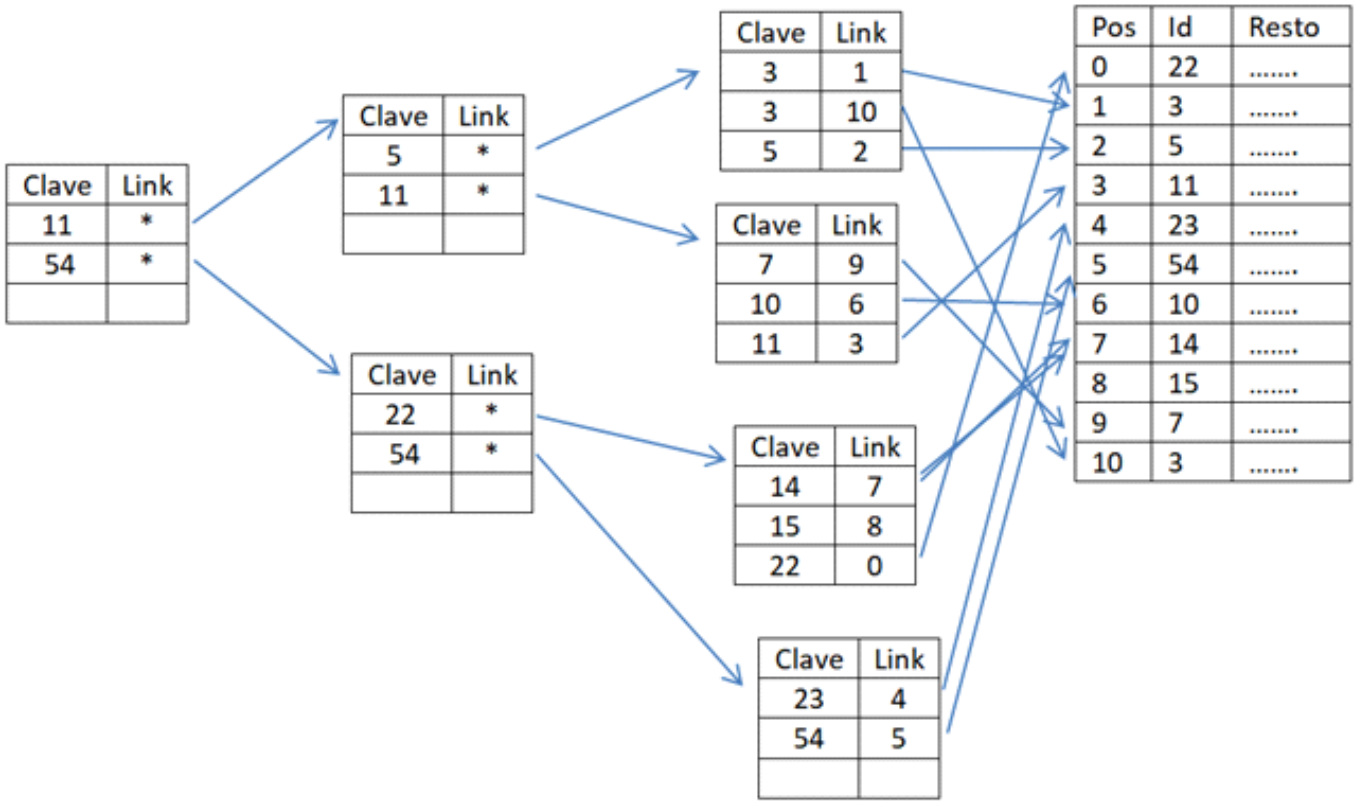
- Se inserta el elemento de **Clave = 15**:



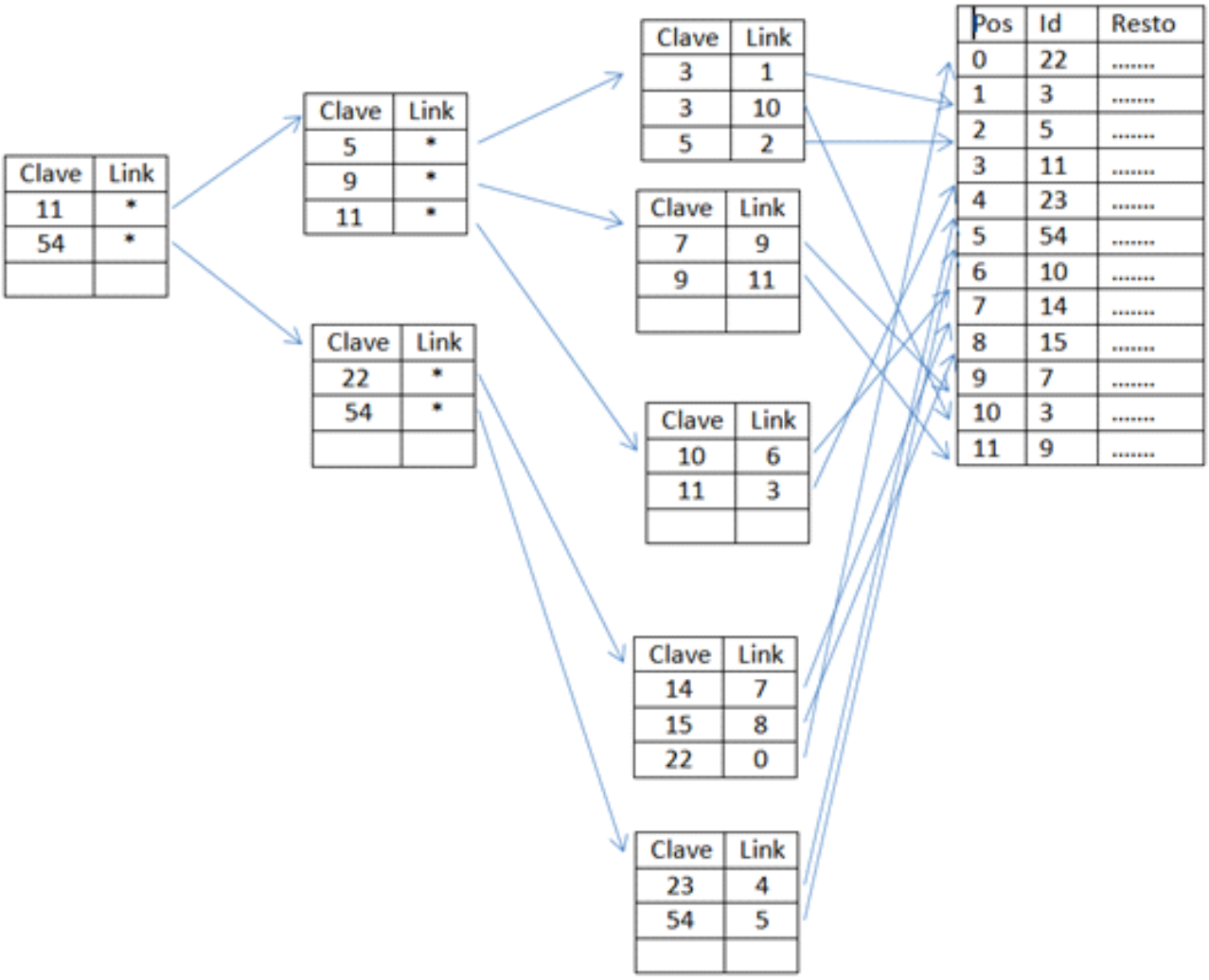
- Se inserta el elemento de **Clave = 7**:



- Se inserta el elemento de **Clave = 3**:



- Se inserta el elemento de **Clave = 9**:


Clave Link
11 *
54 *
Clave Link
5 *
9 *
11 *
Clave Link
22 *
54 *
Clave Link
3 1
3 10
5 2
Clave Link
7 9
9 11
Clave Link
10 6
11 3
Clave Link
14 7
15 8
22 0
Clave Link
23 4
54 5
Pos Id Resto
0 22 .....
1 3 .....
2 5 .....
3 11 .....
4 23 .....
5 54 .....
6 10 .....
7 14 .....
8 15 .....
9 7 .....
10 3 .....
11 9 .....

domingo, 11 de julio de 2021 11:46 a. m.

Unidad 6 - Compresión de Datos

Tipos de Compresión

martes, 29 de junio de 2021 02:52 p. m.

Existen dos tipos de algoritmos de Compresión

- **Con Pérdida**
    - Son algoritmos que **pierden información al comprimir** y que por dicha pérdida **no son reversibles**.
    - Este tipo de algoritmos se utiliza para los **archivos multimedia (imágenes, videos, sonidos, etc)**.
- **Sin Pérdida**
    - Son algoritmos que **comprimen archivos sin perder información**, por lo cual **son reversibles**, dado que permiten volver al estado original del archivo.
    - Se utilizan para **todos los archivos menos los multimedia**.
    - Algunos programas como **WinRAR**, 7zip y otros utilizan algoritmos de compresión sin pérdida.

Como se comprime un archivo

jueves, 1 de julio de 2021 11:21 a. m.

¿Por qué es posible comprimir un archivo?

- El proceso de **compresión** saca provecho de que el **alfabeto ASCII** es de **longitud fija**, entonces se **generará** un **nuevo alfabeto de longitud variable** para reducir el espacio ocupado.
    - Las **computadoras codifican los caracteres mediante ASCII o Unicode**, utilizando un **tamaño fijo** para cada uno de ellos, **1 byte** en el primer caso y **4 bytes** en el segundo.
- En este **nuevo alfabeto** los caracteres **más utilizados o repetidos tendrán una longitud menor** que los caracteres que se repiten con menos frecuencia. Por lo tanto, no todos los caracteres ocuparan el mismo espacio en el alfabeto.
    - *"La l**ongitud variable** es **inversamente proporcional** a la **utilización o repetición** de un carácter."*
    - *"**Cuanto más se utiliza un carácter** deberá tener una **representación más chica en el nuevo alfabeto**."*
- El **alfabeto ASCII** se creó de **longitud fija** de forma tal que **por más que un carácter se repita mucho más que otro el espacio que ocupa es el mismo**.
    - **Todos los caracteres** que lo componen **miden** lo mismo: **8 bits -> 1 byte**
- **No todos los archivos contienen los 256 caracteres especificados en el archivo ASCII**, por lo cual podría ocurrir que **no se utilicen todos desperdiciando** bits en su representación.
- **Una vez comprimido el archivo** este tendrá que tener la **misma cantidad de caracteres**, **los mismos caracteres diferentes** y **en el mismo lugar** en comparación al archivo original.

Algoritmo de Huffman

jueves, 1 de julio de 2021 11:43 a. m.

- **Huffman** es un **algoritmo de compresión** de datos **sin pérdida**, que es **muy eficiente con archivos de texto** (txt, excel, word, pdf, etc).

- El archivo que se quiera comprimir será **recodificado** a través del **algoritmo de Huffman**.

- **Recodificar** el archivo es llevarlo a un **alfabeto diferente** donde **los caracteres no se van a representar o codificar igual que en el alfabeto original**, sino que se van a **codificar en función de sus repeticiones**.

    - Cuanto **más se repite** un carácter **menor representación** va a tener y **cuanto menos se repita** tendrá **mayor representación**.

    - Esto va a generar que el **nuevo alfabeto** tenga **caracteres** que se representan con **2 bits, 3 bits, 4 bits, 12 bits u otras medidas** diferentes porque pasa a ser un **alfabeto de longitud variable.**

- El algoritmo **identifica cada uno de los caracteres distintos** en el conjunto de archivos a comprimir, y le **asigna un código de longitud variable según la frecuencia**.

- Cuanto **más veces aparezca un carácter** en los archivos, la **longitud de su código** va a ser **más pequeña**.

Compresión

jueves, 1 de julio de 2021 12:03 p. m.

A los efectos de simplificar la explicación del funcionamiento del algoritmo, vamos a **mostrar su comportamiento comprimiendo** solo el contenido de un archivo txt que contiene **“EN NEUQUEN”**.

- Se comienza leyendo el archivo e **identificando** todos los **caracteres distintos** que lo componen.

- Esos **caracteres identificados se almacenan conjuntamente con la cantidad de repeticiones** del mismo en un **vector,** este vector se denomina **"tabla de frecuencia"**.

E	N	‘ ‘	U	Q
3	3	1	2	1

- Luego se **ordena el vector por mayor cantidad de repeticiones en forma descendente**, de forma tal que quede en la **primer posición el elemento que más se repite**.

E	N	U	‘ ‘	Q
3	3	2	1	1

- Ya que es necesario obtener una **representación binaria** para cada uno de los **caracteres identificados**, el algoritmo **crea un árbol binario** (ya que este respeta un orden y es binario).

- *"Dividiendo el vector de a dos en función de la cantidad de repeticiones de cada carácter, comenzando con todo el vector como raíz del árbol y llegando hasta que las hojas estén compuestas por un solo carácter."*

- En cada nodo del árbol se encuentran los posibles caracteres a los que se puede acceder y la suma total de la cantidad de veces que se repiten en el archivo los caracteres del nodo.

    - En el caso del nodo raíz se encuentran todos los caracteres ***ENU"Q*** y separado por una coma el numero **10** el cual representa la **suma de la cantidad de repeticiones** de estos caracteres en el archivo original.

    - A través del nodo ***EN , 6*** se puede acceder a los caracteres ***E*** o ***N*** y la **suma de sus repeticiones** en el archivo original es **6**.

- La **creación del árbol** comienza estableciendo el **vector de repeticiones** de los caracteres como el **nodo raíz**.

- Luego **analiza** desde el **principio del vector** si la **cantidad de repeticiones** del **primer caracter** es **igual o mayor que la mitad de las repeticiones del nodo raíz**.

    - En el caso de que la **suma** sea **menor** entonces **analiza el segundo carácter** evaluando la **suma** de **repeticiones del primer y segundo carácter** del nodo.

    - Si la **suma** es **mayor o igual** que la **mitad** de las **repeticiones** del nodo **genera un nuevo nodo** con **estos caracteres** y la **suma de sus repeticiones**.

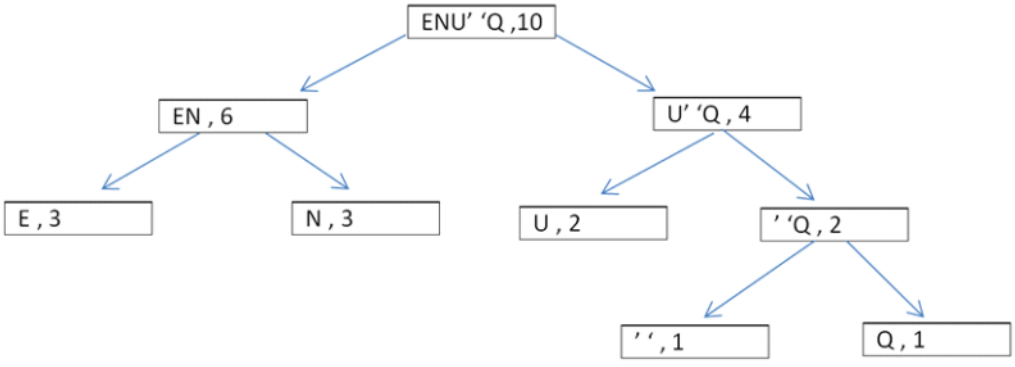
- El **proceso continua generando nodos de esta manera sucesivamente** hasta que en **cada nodo generado** contenga un **solo caracter** con su **respectiva cantidad de repeticiones** en el archivo.

- A **diferencia** de este nuevo árbol, el **árbol** que representa el **alfabeto ASCII** es un **árbol completo y balanceado de 8 niveles,** donde cada una de sus **256 hojas** representaran los **256 caracteres** del **alfabeto ASCII**.

    - El **algoritmo de compresión** busca que el **nuevo árbol** generado sea lo **menos balanceado posible** (cuánto **más desbalanceado** sea el árbol más se podrá comprimir el archivo original).

        - Esto significa que las **hojas** que representan **caracteres repetidos con mucha frecuencia** se hallarán **más cerca de la raíz** (**código** de representación de **longitud menor**)

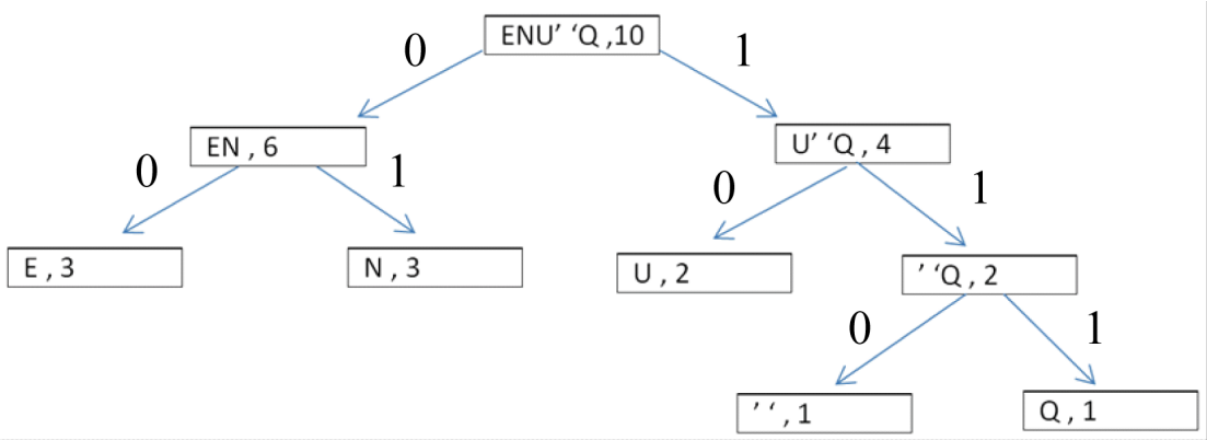
        - Los caracteres **repetidos** con **menos frecuencia** serán **hojas más lejanas a la raíz** (**código** de representación de **longitud mayor**).



Representación de caracteres

jueves, 1 de julio de 2021 12:55 p. m.

- Luego por convención se define que la **izquierda** de cada rama **se representa con 0** y la **derecha** de cada rama **se representa con 1**.



- Se **vuelve a leer el archivo** y para **cada caracter** se va a buscando en el árbol generando el **código de bits** que **representará** ese carácter **en función de que la búsqueda vaya por izquierda o por derecha**.
- Por ejemplo:
  - La **codificación** generada para el caracter **E** del archivo es **00** dado que para acceder a la hoja del árbol que contiene el carácter **E** se debe acceder **dos veces a la izquierda**. Este caracter se **representa** con una longitud de solo **2 bits** ya que es uno de los más repetidos en el archivo.
  - La **codificación** generada para el caracter **Q** del archivo es **111** dado que para acceder a la hoja del árbol que contiene el carácter **Q** se debe acceder **tres veces a la derecha**. Este caracter se **representa** con una longitud de **3 bits** ya que es uno de los menos repetidos en el archivo.
- Al leer todo el archivo se genera la siguiente codificación binaria:

  EN NEUQUEN
  '00:01:110:01:00:10:111:10:00:01'

- Luego se toma esa **cadena de 1** y **0** y se **convierte** en **caracteres** para **generar el archivo comprimido completando** los **caracteres faltantes** indiferentemente con **0** o **1** hasta llegar al **tamaño de byte requerido**.

  [00011100][10010111][10001*000*]

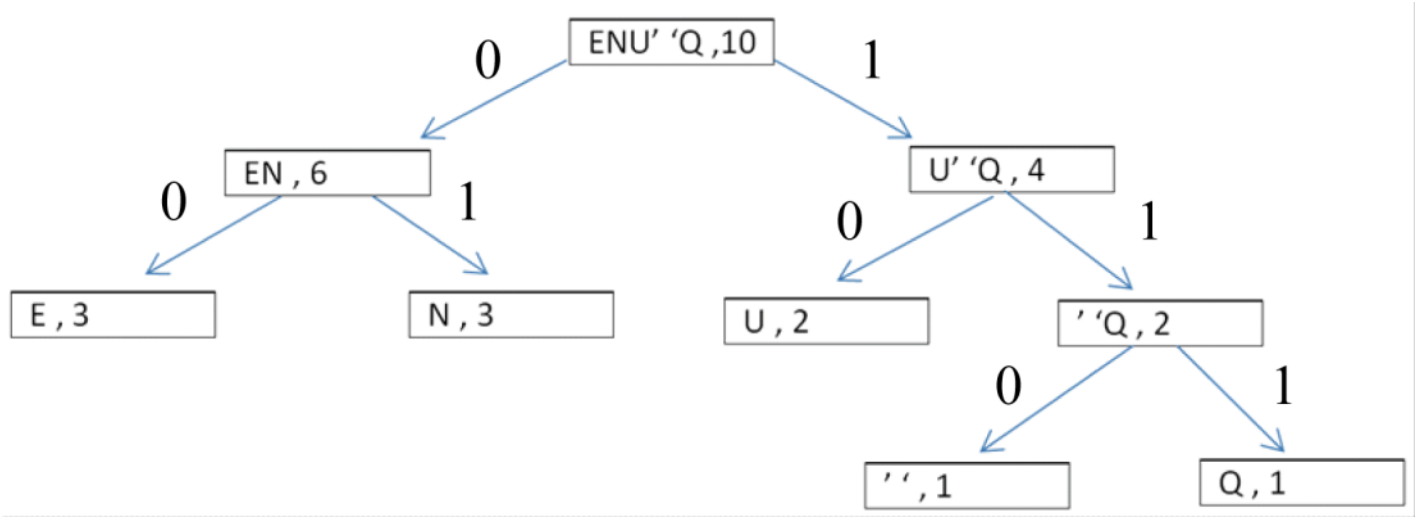
- Ese archivo generado es el **archivo comprimido.** El **archivo original** se representaba con **10 bytes** y el **archivo comprimido** se representa en solo **3 bytes**.

Descompresión

jueves, 1 de julio de 2021 01:37 p. m.

Para **descomprimir** el **archivo generado** se parte del archivo comprimido y se realiza la **lectura** del mismo para que **en función** del **nuevo árbol generado** se **obtengan los caracteres originales**.

’00:01:110:01:00:10:111:10:00:01’
[00011100][10010111][10001*000*]



Generando EN NEUQUEN

- Observación: **Una vez recuperados los caracteres necesarios se deja de leer la cadena de 0 y 1 generada** aunque no la haya leído completamente **(en algunos casos sobran caracteres, representados en rojo anteriormente)**. El **proceso** de **descompresión** sabe **cuántos caracteres debe recuperar**.

Consideraciones a tener en cuenta

jueves, 1 de julio de 2021 01:46 p. m.

Comparación entre tiempo de compresión y descompresión de un archivo

- El tiempo de compresión de un archivo será mayor que el tiempo de descompresión ya que para comprimir se debe leer el archivo original una vez para generar el vector de frecuencias de caracteres utilizados y una segunda vez para generar un código de bits distinto a cada carácter según el árbol generado.
- En cambio el proceso de descompresión lee una vez el archivo comprimido para procesarlo y graba el contenido del archivo original donde se estuviese descomprimiendo.
- En comparación el proceso de compresión realiza una mayor cantidad de accesos a memoria que el proceso de descompresión.

El mismo proceso de compresión es más eficiente en archivos de textos que en otros como archivos ejecutables, librerias u otros que no sean de texto

- Al generarse por ejemplo un archivo ejecutable la distribución de los caracteres repetidos tiende a ser una distribución normal que la distribución de los repetidos en un archivo de texto.
- En un archivo de texto se repiten mucho las vocales por ejemplo y eso permite comprimir mucho el archivo, en cambio en un archivo ejecutable los caracteres no se repiten tanto.

Se almacena el vector con los caracteres y su cantidad de repeticiones

- El **vector** con los **caracteres** y su **cantidad de repeticiones** ordenado se guarda en el **archivo comprimido** por lo que **no se comprimen archivos muy pequeños**, en esos casos el vector ocuparía más espacio el archivo comprimido.
    - En el peor de los casos (256 caracteres distintos y con su cantidad de repeticiones) el **vector** puede pesar **2 KB** aproximadamente entonces **no se comprimen archivos de ese tamaño** o menores.
- El **árbol** no hace falta guardarlo en el archivo ya que es una **abstracción** que se genera a partir del **vector** y no tiene sentido crearlo en memoria por **ocupación de memoria** y **pérdida de tiempo.**

Breve explicación del algoritmo

- La **representación** y **recuperación** de caracteres se realiza con un a**lgoritmo simple de FOR e IF** analizando al recorrer el vector que **carácter debe representar o retornar**, por eso no es necesario almacenar el árbol como estructura de datos literalmente.
- A continuación desarrollamos el **algoritmo de recuperación**, el **algoritmo de representación** es muy similar.
    - Para **recuperar** un carácter se recorre el vector hasta la "mitad" (donde se encuentra la mitad del total de repeticiones) y se fija con los valores leído que mitad del vector deberá utilizar para volver a iterar el mismo algoritmo hasta llegar a un nodo especifico.

E	N	U	‘ ’	Q
3	3	2	1	1

    - Por ejemplo: si se lee **00**, la primera iteración se ubica en la "mitad" del vector y como tiene un **0** vuelve a llamar al algoritmo pero con la **mitad izquierda** del **vector.**

E	N
3	3

    - Luego se vuelve a situar en la mitad de este nuevo vector y como tiene un **0** vuelve a llamar al algoritmo pero con la **mitad izquierda** del **vector.** En este caso la mitad izquierda es el nodo ***E , 3*** entonces se lee el carácter **E**.

E
3

Determinar donde finaliza el vector y donde comienzan los 0 y 1 generados que representan caracteres

- En el **archivo comprimido** se almacenan el **vector** y **seguidamente** los **caracteres** que corresponden al contenido del archivo original, todo se expresa con un **conjunto de 0 y 1 sin delimitación**.

- Se determina **donde finaliza el vector** a través del **header** del archivo. Las distintas **"extensiones" de archivo** como por ejemplo ***.doc*** tienen asociadas un **header** con **cierta estructura** que **especifica donde termina el vector** y **comienzan** los **0** y **1** generados.

Identificar caracteres EOF que pueden generarse en posiciones al azar al comprimir un archivo

- Los caracteres de escape están representados por 2 caracteres concatenados (como '\0') porque si se utilizara alguno de los caracteres existentes en ASCII de manera individual este quedaría inutilizado.

- En la computación solo hay un solo tipo de archivo, archivos binarios o de texto es solo una abstracción, lo que cambia es la forma de abrirlos a la hora de leerlos o escribirlos (modo binario o texto), ya que la computadora solo lee y escribe 1 o 0 sin distinguir en que archivo específicamente.

    - Modo binario
        - El archivo no interpreta caracteres especiales o de control ni secuencias de escape.
            - No interpreta caracteres como '\n' solo los mostraría.

    - Modo texto
        - El archivo interpreta caracteres especiales o de control y secuencias de escape.
            - Debe interpretar caracteres de control como '\n' para mostrar el salto de línea.

- Cuando se lee un caracter con ***fget***() esta función lee y devuelve "algo" del tipo **int** ya que de esta manera devuelve 2 bytes compuesto por los High Order Bits y Low Order Bits, como por ejemplo 01011110 10001101.

- Para detectar correctamente caracteres de EOF, se leen de a 2 caracteres y si el conjunto coincide con un EOF se termina la lectura del archivo, en caso contrario toma el High Order Bits individualmente como lectura y analiza el Low Order Bits con su siguiente Byte.

- Con la representación resultante del archivo comprimido puede ocurrir que sin intención se concatenen 2 caracteres que representen un EOF, esto provocaría que al leer el archivo abriéndolo en modo texto se interprete EOF donde no se debería.

Compresión Multimedia

jueves, 1 de julio de 2021 02:00 p. m.

Este tipo de archivo se comprime a través de algoritmos de compresión Con Pérdida.

- Estos algoritmos de compresión **recodifican** la **resolución** o la **definición** de un archivo multimedia, nunca ambas cosas a la vez.
    - Por ejemplo: Si se recodifica la definición de un archivo multimedia de *True Color 64 bits* a *True Color 32 bits* se reduciría a la mitad el tamaño del archivo ya que se almacena la mitad de bits para representar el color de cada pixel.

La forma de comprimir archivos multimedia es modificar su codificación:

- **Recodificando** la **codificación** de ***resolución*** de los archivos **bajando la calidad** de los mismos para **utilizar menos bits para su representación**.
    - Resolución: cantidad de pixeles activos.
- **Recodificando** la **codificación** de ***definición*** de los archivos **bajando la calidad** de los mismos para **utilizar menos bits para su representación**.
    - Definición: En imágenes representa la cantidad de colores distintos que se pueden utilizar para representar un pixel.
        - True Color 64 bits: 8 bytes por color
        - True Color 32 bits: 4 bytes por color

domingo, 11 de julio de 2021 11:46 a. m.

Unidad 7 - Integridad y Encriptación

Integridad

- **Integridad**
  - Es el **proceso** que permite **identificar la integridad de un archivo**, o sea, poder **validar que un archivo no ha sufrido cambios ante un proceso que lo modifico**, sea este por una **transferencia** o un **proceso de cambio** como la **compresión** y **descompresión** del mismo.
    - **Por ejemplo**: cuando se transmite mediante una red o se copia o se le realiza un proceso de compresión y descompresión aparece la necesidad de verificar si el nuevo archivo obtenido es igual al original.
  - *Cuando se produce una modificación en un archivo a través de un proceso modificatorio se requiere conocer la integridad del nuevo archivo generado.*

  - Si bien es cierto que la única forma de verificar que un archivo es igual a otro requeriría una lectura de ambos archivos con una comparación caracter a caracter, existen medios tecnológicos para poder validar la veracidad del archivo sin necesidad de contar con el archivo original.
  - Existen otros métodos más eficientes y de alta fiabilidad los cuales no necesitan comparar el archivo destino con el archivo origen para comprobar la integridad una vez realizado el proceso.

Control de Integridad

sábado, 3 de julio de 2021 04:22 p. m.

Que se debe controlar para establecer la igualdad de los archivo Origen y Destino

- **Tamaño**
  - Ambos archivos deben tener el **mismo tamaño en cantidad de caracteres**.
- **Contenido**
  - Ambos archivos deben contener los **mismos caracteres**.
- **Posición**
  - Considerando que ambos archivos contienen los mismos caracteres, **dichos caracteres deben estar en la misma posición**.

CHECKSUM

sábado, 3 de julio de 2021 04:28 p. m.

Se utiliza un **polinomio** para validar **tamaño**, **contenido** y **posición** del archivo destino.

- **Tamaño** = **grado** del polinomio.
  - El **tamaño** está dado por el **grado del polinomio**.
- **Contenido** = **coeficiente** del polinomio.
  - El **contenido** está dado por los **coeficientes del polinomio**.
- **Posición** = **potencia de la x** del polinomio.
  - La **posición** está dada por el **grado** que acompaña a la **X del polinomio**.

- **Proceso del método CHECKSUM**
  - Antes de perder el **archivo original** el método crea un **polinomio** que representa **tamaño, contenido** y **posición** del archivo original. Luego genera **otro polinomio** a partir del **archivo destino** una vez realizada la modificación o proceso.
  - El método se llama **"CHECKSUM"** porque **chequea** que la **suma** resultante con el **mismo valor de raíz** en los **polinomios** generados a partir del archivo original y el archivo destino **sea igual**.

Procedimiento (ej. Archivo que contiene la expresión 'HOLA'):

1. Calcula un polinomio: $Hx^0 + Ox^1 + Lx^2 + Ax^3$
2. **Asigna un valor arbitrario a x** para resolver el polinomio.
3. El resultado se agrega al archivo como **redundancia**, llamado **Hash**.
4. Realiza el proceso de transformación.
5. Genera un nuevo polinomio y obtiene el valor.
6. **Chequeo de redundancia cíclica (CRC)**: compara el valor nuevo con el anterior.

Ejemplo CHECKSUM

sábado, 3 de julio de 2021 04:44 p. m.

Tomemos a modo de ejemplo que se quiere validar un archivo que contiene la expresión **'HOLA'**

Para validar esta situación se puede armar un **polinomio** donde se utilicen los caracteres que componen el archivo como coeficientes del polinomio:

$$\boldsymbol{Hx^0 + Ox^1 + Lx^2 + Ax^3}$$

Antes de realizar el **procedimiento** del archivo se **calcula el polinomio** y se **resuelve** aplicando una **raíz específica**.

Luego el **resultado se agrega al archivo destino** generado vía el **proceso** de transformación.

Luego se vuelve a **generar un polinomio** con el **contenido** del **archivo destino**.

Se **resuelve** dicho **polinomio** y se **compara** el **resultado obtenido con el almacenado en el archivo**, si es **igual** se puede afirmar que **los archivos son iguales**.

Se puede dar el caso que los **polinomios generados** a partir del **archivo original** y el **archivo destino** tengan **coeficientes distintos** pero que al resolverlos utilizando el **mismo valor de raíz** den el **mismo resultado**.

Esto se da porque el **ámbito combinatorio de caracteres ASCII es muy amplio**. La **probabilidad** de que esto ocurra es **remota**, pero si ocurriera provocaría que el **método CHECKSUM** arroje un **valor erróneo de integridad**.

En la **realidad NO se arma un polinomio con caracteres como coeficientes**, sino que se toman los **bits que componen dicho carácter** de forma tal de **iluminar** o **apagar potencias** para **evitar las fallas** en el método.

Si **H** se representara como **00110011** el polinomio se armaría así:

$$\boldsymbol{0x^0 + 0x^1 + 1x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 1x^7}$$

De esta forma la **posibilidad de error** de la función **es nula** porque las diferencias son incompensables intercambiando potencias, las potencias no son combinables.

> El **error es nulo** porque **no pueden compensarse las potencias**, no se puede por ejemplo, apagar x^3 y prender x^1 y x^2 y que me resulte el mismo valor.

El **valor de raíz** que se toma es un **valor** en el intervalo **comprendido** en el **rango (0:1)**, **no** se puede **evaluar** los polinomios con los **valores 0 y 1** ya que **anularía** las **potencias**.

> El **valor** asignado a la **raíz entre 0 y 1** para que crezcan en mantisa los valores del polinomio, no crezcan en parte entera para que su representación sea más sencilla ya que la computadora utiliza menos espacio para representar un valor Double (decimal) que para uno Long (enteros).

CRC

sábado, 3 de julio de 2021 05:30 p. m.

Control de redundancia cíclica (CRC)

- **No se toma todo el contenido del archivo**, sería muy pesado y no podría calcular el tamaño.
- **Toma caracteres por ciclo para no verificar caracter a caracter.**
    - El método siempre evalúa el primer y último carácter de los archivos, entre ellos genera ciclos para evaluar otros caracteres.
        - Por ejemplo: el primero, uno del medio y el último.
    - La cantidad de ciclos se determina en base a la posibilidad de error que se pueda permitir considerando que a mayor cantidad de ciclos menor es la posibilidad de error pero más tiempo tardará el proceso.
    - **nota**:
        - Si el método da bien, se puede asegurar que todo el archivo es correcto.
        - Como el método trabaja a nivel bits un error en el medio se trasladaría y los demás caracteres también cambiarían. Por este motivo no es necesario analizar caracter a caracter.
- **CRC varía la cantidad de caracteres** que evalúa, de 32bits a 128bits, mientras más bits menor posibilidad de error.
    - **nota**:
        - La cantidad de bits se elegirá en base a las características del procedimiento que está realizando ya que hay procedimientos que tienen más riesgo de error.
        - Por ejemplo, un proceso de compresión y descompresión tiene un riesgo de error pero si otro procedimiento aparte de comprimir el archivo lo envía a través de la red a otro punto para luego descomprimirlo el riesgo de error es más alto. Para este último deberá utilizarse un CRC de menos cantidad de bits para evaluar su integridad.

Encriptación

sábado, 3 de julio de 2021 04:13 p. m.

- Es el **proceso** que permite **modificar el contenido de un archivo** para que **mantenga el contenido establecido**, pero que **el mismo no pueda ser visible en un formato tradicional** del mismo.
- El **objetivo** es **ocultar la información** contenida en el archivo para que **no pueda ser legible**.
- El **archivo** debe ser **modificado sin cambiar su tamaño y espacio ocupado** a tal fin.
- Para **encriptar** existen **innumerables métodos** para realizarlo, nos concentraremos en las diferentes **formas** de encriptar un archivo que existen.
- **Nunca** se encripta **TODO** el **contenido** del archivo sino que **se encriptan algunas partes**. Si se encriptara la totalidad del archivo para poder obtener el contenido original se debería desencriptar la totalidad del archivo también, **generando grandes pérdidas de tiempo y recursos**.

Procesos de Encriptación

- **Desplazamiento**
  - Los procesos de encriptación se basa en el **desplazamiento** de los **caracteres** en **función de algún patrón.**
- **Reemplazo**
  - Los procesos de encriptación se basan en el **reemplazo** de **determinados caracteres** en **función del algún patrón** donde ese reemplazo **puede ser fijo o variable**, con lo sin intervención del usuario
- **Mixto**
  - Se aplican **ambos procesos** en **cualquier orden**, **reemplazo** y **desplazamiento**.

Por Reemplazo

sábado, 3 de julio de 2021 06:05 p. m.

- **Reemplazo Fijo**
    - Se eligen **posiciones** por un patrón y se **reemplazan los caracteres** de esas posiciones por un **valor definido por el algoritmo**.
    - Para **desencriptar** este tipo de archivos es **necesario conocer solo el algoritmo de encriptación**, de otra manera es imposible obtener el contenido original del archivo.
    - **Por ejemplo**, a los caracteres que se encuentran en posiciones pares se les suma 8, a los impares 15, a las potencias de 6 se les suma 23, etc.
- **Reemplazo Variable**
    - Se **encripta** el archivo con una **clave dispuesta por el usuario** y dicha **clave** se **copia** al **contenido del archivo** o **valor a encriptar**.
    - Para **desencriptar** este tipo de archivos es **necesario conocer el algoritmo de encriptación pero también la clave dispuesta por el usuario**, ya que el archivo original fue encriptado en base a esa clave.

Por Desplazamiento

sábado, 3 de julio de 2021 06:06 p. m.

- *Se **desplazan valores** en forma similar a lo que se conoce como "sopa de letras" pero en lugar de realizarlo por caracteres **se realiza por bits y no a nivel byte**, modificando totalmente el contenido del mismo.*
    - **Desplazamiento** de los caracteres **en función de algún patrón.**
    - Se realiza **a nivel bit** para **cambiar totalmente el contenido** del mismo.
    - Por ejemplo: intercambiar los bits pares por los impares excluyendo múltiplos de 7.

Encriptación Mixta

sábado, 3 de julio de 2021 06:06 p. m.

Se aplican **ambas formas de encriptación** una primero y otra después, **en cualquier orden de ejecución**, de forma tal de dar **mayor seguridad a la encriptación**.

domingo, 11 de julio de 2021 11:46 a. m.

Unidad 8 - Business Intelligence

Inteligencia de Negocio

domingo, 4 de julio de 2021 12:15 p. m.

Modelo Transaccional

- **Inicialmente** el modelo era **OLTP**, Online Transaction Processing, **únicamente procesando transacciones** para obtener el resultado esperado.

Modelos de inteligencia de negocio

- **Inteligencia de Negocios**: **conjunto de metodologías, herramientas y estructuras de almacenamiento** que permiten la **reunión, depuración y transformación de los datos** en una **información integrada** que se pueda **analizar y convertir en conocimiento** para la **optimización** del proceso de **toma de decisiones**.
- Herramientas **OLAP transforman datos en información**, brindan **información** para la **toma de decisiones**.

DATOS, INFORMACION Y CONOCIMIENTO

- Estos términos **se consideran sinónimos** pero entre ellos **existen grandes diferencias**
    - Los **datos** son **valores ya conocidos** que se encuentran **diseminados en diferentes partes.**
    - La **información** es un **dato asociado a una relación**
    - El **conocimiento** es **información** que **vinculada y acumulada** luego de un **proceso de análisis** se transforma en **conocimiento**.

Datos

domingo, 4 de julio de 2021 02:10 p. m.

- Los datos son la **mínima unidad semántica** que se corresponden con los **elementos primarios o absolutos** (un número, una palabra, etc.) de la información que **en sí mismos no tienen ningún valor**.
    - Para brindar algún tipo de información necesitan que se los vincule con alguna relación.
- También se pueden tomar como un **conjunto discreto de valores absolutos que no pueden aportar nada que contribuya con la toma de decisiones**.
    - Por ejemplo: un número, un nombre, etc. Asimismo, un CD, un DVD o un disco rígido pueden almacenar físicamente una colección de datos.
- Los datos, en general, provienen de diferentes orígenes: internos; es decir, de la propia organización, o externos, extraídos del contexto; por ejemplo, de la competencia. A la vez, pueden ser objetivos, subjetivos y de tipo cualitativo o cuantitativo.

Información

domingo, 4 de julio de 2021 02:13 p. m.

- Se puede definir a la **información** como el **conjunto de datos relacionados** con un **significado específico**.

- En consecuencia, **si se le añade alguna relación**, los **datos** se pueden **convertir en información**.

- A continuación, se describe **de qué manera un dato** se puede **transformar** en **información**:

    - **Contextualizando**: se sabe en qué **contexto** y para qué **propósito** se generaron.
    - **Categorizando**: se conocen las **unidades de medida** que ayudan a interpretarlos.
    - **Calculando**: los datos fueron **procesados matemática o estadísticamente**.
    - **Corrigiendo**: habiendo **eliminando errores** o **inconsistencias** de los datos.
    - **Condensando**: **resumiendo** los datos de **forma más concisa** (agregación de datos).

Conocimiento

domingo, 4 de julio de 2021 02:15 p. m.

- El **conocimiento** es la **fusión de valores** entre **información** entrante y **experiencia obtenida anteriormente.** A medida que se va **incorporando más información** se **generan nuevos conocimientos** que contribuirán con la **toma de decisiones**.

- Para que la **información se convierta en conocimiento** se deben llevar adelante las siguientes acciones:

    - **Comparación con otros elementos**

        - Por ejemplo comparando inscriptos de utn sistemas con los de uba sistemas me doy cuenta que aumentó el interés en un campo.

    - **Predicción de consecuencias**

        - Por ejemplo analizando el aumento de alumnos en sistemas puedo analizar consecuencias a futuro.

    - **Búsqueda de conexiones.**

        - Por ejemplo dentro de la gente que entro a ingeniería nueva el 50% es de sistemas, por qué? por la demanda del mercado por ejemplo.

    - **Conversación con otros portadores de conocimiento.**

        - Por ejemplo la "conversación" en una clase entre alumnos y un profesor.

OLAP vs OLTP

domingo, 4 de julio de 2021 02:32 p. m.

- La **aparición** y **evolución** de la **informática** a finales de la **década de los 90** trajo una **acumulación masiva de datos** en el entorno empresarial. Esta **gran cantidad de datos proviene** en su mayor parte de la **aplicación de la informática** en las **actividades transaccionales** de la empresa, contabilidad, gestión de almacén, facturación.

- H**ace falta un modo de estructurar la información y los datos** que aporte una **nueva perspectiva** de los mismos.

- Nace de este modo la tecnología **O.L.A.P. "On Line Analytical Processing"** basada en la utilización de **tecnología de Bases de Datos Multidimensionales**, para diferenciarse de **OLTP "On Line Transaction Processing"** que se fundamentan en **Bases de Datos Relacionales**.

Conceptos

- **OLTP**: **"On Line Transaction Processing"** también llamado **Modelo Transaccional** debido a que se basa en la **ejecución** de un **conjunto de transacciones** para obtener el resultado esperado.

- **OLAP**: **"On Line Analytical Processing"** también llamado **Modelo Relacional** debido a que **analiza** y **relaciona** la **información analizada**.

    - A diferencia de las transacciones **OLTP** las transacciones **OLAP** procesan información analizándola, generando conocimiento para ayudar en la toma de decisiones.

        - Por ejemplo: Si una persona lee todo el contenido de un libro estaría leyendo renglón tras renglón como procesando simples transacciones (OLTP), en cambio si la persona lee todo el libro pero con la finalidad de realizar un resumen del mismo, entonces se lee renglón tras renglón pero con el fin de analizar toda la información obtenida posteriormente para realizar el resumen (OLAP).

OLTP

domingo, 4 de julio de 2021 02:39 p. m.

Características

- Su **ejecución** se basa en **transacciones**.
- Conforman el **99%** de los **sistemas existentes**.
- Son sistemas "**operativos**": **operan o resuelven** una **operación específica**.
  - Un sistema de compras por ejemplo procesa y opera compras.
  - El sistema no piensa ni analiza, solo opera.
  - Utilizados por el nivel "operativo" de la organización
    - (nivel más bajo de la pirámide de toma de decisiones)
- Procesan **datos**
  - Estos procesos reciben simples números, nombres, etc. No ingresan datos relacionados.
- Los **datos** se almacenan **normalizados**.
- **Registran dato**s nivel de **detalle de cada transacción**.
- Los **datos volátiles**, **cambian en el tiempo** por la ejecución de transacciones.

Uso

- Las **aplicaciones** con **OLTP** están caracterizadas en que **muchos usuarios crean, actualizan, o retienen registros individuales**. Entonces, las **bases de datos con OLTP** son **optimizadas** para las **actualizaciones de las transacciones**.

OLAP

domingo, 4 de julio de 2021 02:43 p. m.

Características

- Su **ejecución** se basa en el **análisis** de la **información que genera** el modelo **OLTP.**
- Conforman el **1%** de los **sistemas existentes**.
- Son **sistemas** para la **toma de decisiones.**
  - Son utilizados por el **nivel directivo** de la **organización** el cual se encarga de la **toma de decisiones**.
- Procesan **información**. **Datos recolectados y relacionados** previamente.
- La **información** se **almacena desnormalizada**.
- **Registran información global por patrones de interés** también conocidos como "**dimensiones**".
  - Patrones de interés en una facultad por ejemplo: alumnos, cursos, materias, etc.
  - No interesa saber la información exacta de cada uno individualmente, sino que se globaliza según alguno de los patrones de interés.
- La **información** es **persistente** o **"no volátil"**, se alimenta de un sistema transaccional.

Uso

- Las **aplicaciones** en **OLAP** son **usadas por analistas y gerentes** que frecuentemente quieren **vistas de alto nivel de los datos**, tales como las **ventas de una línea de productos, por región, etc**.
- La **base de datos OLAP** es usualmente **actualizada por bloques**, generalmente de **múltiples fuentes**, y **provee poderosas aplicaciones multiusuario de poder analítico**.
  - Por lo tanto, las **bases de datos OLAP** son **optimizadas** para el **análisis**.
- Las **herramientas OLAP** que se han ido desarrollando **han buscado mantener una compatibilidad con otras herramientas de análisis que ya se utilizaban** en la empresa, como son las **hojas de cálculo** tradicionales.
- La mayoría de los usuarios coincide en la **facilidad y beneficio de utilizar aplicaciones multidimensionales** pero presentan una actitud expectante ante los **problemas de construcción y mantenimiento**.

Estructura OLAP

domingo, 4 de julio de 2021 03:26 p. m.

- La **gran mayoría** de los **datos** que se usan en aplicaciones OLAP son **originarios de otros sistemas y aplicaciones**.

- De cualquier modo, en casi la **totalidad de las aplicaciones OLAP**, los **datos son capturados directamente por la aplicación OLAP**.

- Los **datos** que proceden de otras aplicaciones es **necesario duplicarlos y almacenarlos separadamente de los originales** de los cuales proceden, para **poder ser utilizados de manera activa** por la aplicación OLAP de **manera independiente**.

Motivos de Duplicación

domingo, 4 de julio de 2021 03:32 p. m.

Algunas de las **razones que obligan a duplicar los datos** para formar el **modelo OLAP** son:

- **Ejecución**
    - Las **aplicaciones OLAP** son con cierta frecuencia de un **gran tamaño** y se suelen usar para realizar análisis interactivos inciertos.
    - Esto requiere que se pueda **acceder a los datos de manera muy rápida**, lo cual obliga a que **se guarden separados**, y a disponer de una **estructura de datos optimizada** que pueda ser **accedida sin perjudicar la respuesta operativa del sistema**.

- **Múltiples fuentes de datos**
    - Para realizar consultas sobre **datos provenientes de múltiples fuentes diferentes sin tener que interactuar con múltiples sistemas**, los cuales tienen distintas codificaciones y periodicidades.
    - Si un **sistema OLAP** se alimenta de **OLTPs diferentes**, es mejor acceder a **una única fuente** que a múltiples.

- **Filtrado de datos**
    - En los **sistemas transaccionales** nos encontramos con **gran cantidad de datos** que necesitan ser filtrados antes para poder realizar un buen análisis que nos permita generar informes adecuados para no tomar una decisión equivocada.
    - En el **modelo OLAP** se **almacenan** los **datos pre-filtrados** para poder **consultarlos directamente** a la hora de hacer un análisis que permita tomar decisiones. Lo cual **agiliza** los **procesos de consulta y análisis** de estos.

- **Ajuste y modificación de datos**
    - Hay varias razones por la cuales **los datos deben ser ajustados antes de realizar el análisis**, como por ejemplo:
        - **Sucursales situadas en otros países** operan con **contabilidades distintas** y los datos puede que necesiten ser modificados antes de usarse en el análisis.
        - Las distintas **estructuras** de la compañía **no siempre son iguales**. Existe **diferencias** en los **modos de trabajar** de las direcciones departamentales, en las estructuras operativas, etc.
        - Se pueden realizar análisis que **no parten de datos operativos** como pueden ser por ejemplo los datos que se obtienen de las características demográficas.

- **Actualización y consistencia de datos**
    - **Actualizar datos en cortes periódicos** para **no ser afectado por los diferentes períodos** de actualización de las fuentes, ya que el análisis que realiza un **OLAP depende** en gran medida de la **consistencia de los datos**. Por lo tanto **es necesaria una plataforma que garantice esa consistencia**.
    - Si los **datos** que van a ser **usados** por una **aplicación OLAP** proceden de **distintas fuentes** es muy probable que **no se actualicen todos al mismo tiempo**.
    - Es decir, **las aplicaciones** de las cuales **proceden los datos** pueden estar, y de hecho seguro que ocurrirá así, en **diferentes estados de actualización**.

- **Historia de los datos**
    - La gran mayoría de **aplicaciones OLAP** incluyen el **tiempo como una dimensión**.
    - El **uso del tiempo** como una **dimensión permite** obtener **resultados muy provechosos** en cuanto a **análisis temporales** cuando se dispone de datos de varios años atrás.
    - La adquisición de datos de años anteriores supone un importante esfuerzo, ya que **es necesario migrar estos datos de aplicaciones antiguas** y **ajustarlos** para que puedan ser utilizados en la base de datos del OLAP.

- **Distintas perspectivas o vistas**
    - **Observar patrones de interés** en vez de datos operacionales detallados. Para ello interesa combinar almacenes de datos, **ajustar la información según** el **nivel de resumen** o el **nivel de visión** que se quiere alcanzar.
        - **Patrón de interés: sujeto para el cual** (o sobre el cual) **se quiere obtener información**. En base a ellos se agrupa la información.

- **Actualización de datos**
    - Si la **aplicación** dispone de **varias entradas de datos** es **necesario separar** la **base de datos de OLAP** para que **no se sobreescriban los datos operacionales** que se están usando en un determinado momento.

Base de datos Multidimensionales

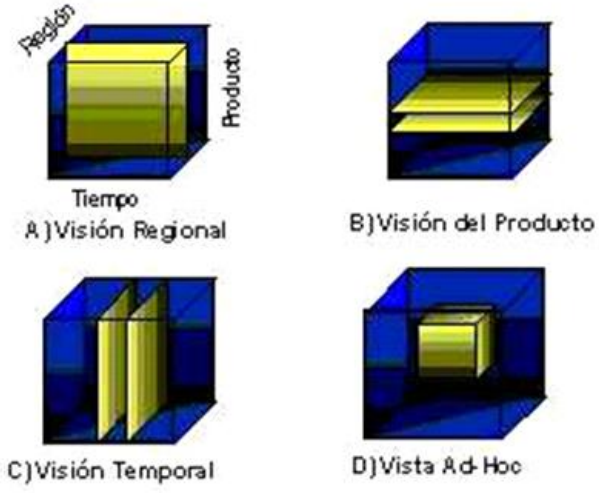
domingo, 4 de julio de 2021 04:08 p. m.

- Modelo basado en **N dimensiones**, guardando la **información en la intersección de las dimensiones**.
  - nota: en lugar de hablar de filas y columnas se habla de dimensiones.

- Las **dimensiones** determinan la **estructura de la información almacenada** y **definen adicionalmente caminos de consolidación**.

- La **información almacenada** se **presenta como variables** que a su vez están **caracterizadas por una o más dimensiones.**

- De este modo la **información** puede **analizarse dentro del cubo formado por la intersección de las dimensiones** de la variable particular.



Dispersión de datos

domingo, 4 de julio de 2021 04:15 p. m.

- A medida que se **agregan dimensiones** a una **Base de Datos Multidimensional**, el **número de puntos de datos o "celdas" crece rápidamente**.

- Por ejemplo, considerando que no se venden todos los productos en todas las sucursales todos los días, si se considerara que las sucursales más pequeñas solo pueden manejar el 20% de todos los productos, el 80% de las celdas estarán vacías.

- En la práctica, muchas Bases de datos tienen el 95% de las celdas vacías o en cero. Esto es conocido como "sparsely populated", "poblados dispersos", "dispersión de datos" o simplemente sparse.

- **Hipercubo**

    - Información guardada en **un único cubo**, estructura multidimensional sencilla.

        - **Estática.**
        - **Más rápido, acceso directo.**
        - **Ocupa más espacio.**
        - **nota: técnicamente es un vector multidimensional.**

    - La **información se guarda implícitamente en un gran y único cubo**, presentando los datos al usuario en un formato de hipercubo, donde todos los datos en la aplicación aparecen como una **sencilla estructura multidimensional**.

- **Multicubo**

    - La información se almacena en **múltiples objetos multidimensionales más pequeños y densos**.

        - **Dinámica.**
        - **Más lento, no tiene acceso directo.**
        - Saltea los elementos vacíos, apunta directamente al siguiente lleno.
        - **Disminuye la dispersión de datos**.

    - La información se almacena **dividiendo** los **datos en grupos más pequeños y densos** (**objetos multidimensionales**), donde la base de datos multidimensional consiste en un número de objetos separados normalmente con diferentes dimensiones.

    - Se **evita bastante la dispersión de datos** pero es mucho más lento porque requiere de muchos más accesos que no son directos a diferencia de un subíndice como es en un hipercubo, en un multicubo se accede a otra estructura de datos.

BDM vs BDR

domingo, 4 de julio de 2021 04:24 p. m.

	Base de Datos Relacional	Base de Datos Multidimensional
Deposito de datos, acceso y visión	Relacional Tablas de Columnas e hileras Lenguajes SQL con ampliaciones Herramientas de terceros que usan API	Dimensional Arreglos: Hipercubo, Multicubo Tecnologia de matriz dispersa Propietario de hoja de calculo
Utilización e incorporación	OLTP Motor RBDMS Profundización a nivel de detalle Desempeño de consultas: rango amplio	OLAP Motor multidimensional Profundización a nivel de resumen/adición Desempeño de consultas: rápido
Tamaño y actualización de bases de datos	Gigabytes a terrabytes El deposito de indices y el retiro de normas que incrementan tamaño Consulta y cargas paralelas Actualizacion durante uso	Gigabytes Compresion y adicion de datos dispersos Dificil actualizar durante uso; los cambios pequeños pueden requerir reorganizacion

---

domingo, 4 de julio de 2021 04:43 p. m.

Data Warehouse (DW)

domingo, 4 de julio de 2021 04:43 p. m.

- **Data Warehouse (DW)**
    - *Es una base de datos corporativa cuya característica principal es la integración y el filtrado de información de una o varias fuentes, que luego procesará para su análisis desde diferentes punto de vista y con una gran velocidad de respuesta.*
    - *Es una colección de datos históricos e integrados diseñada para soportar el procesamiento informático para la tomas de decisiones estratégicas que no utilizan para la operatoria diaria.*
- A partir de la década del '90 los **sistemas de Data Warehouse** se convirtieron en el centro de la arquitectura de los **sistemas de Información** y surgieron para **resolver los problemas que acarreaba la extracción de información sintética desde datos atómicos** almacenados en **bases de datos de producción**.
- Un sistema de **Data Warehouse** incluye las siguientes **utilidades**:
    - **Integración de bases de datos heterogéneas** (relacionales, documentales, geográficas, archivos, etc.).
    - Ejecución de **consultas complejas no predefinidas** que visualicen el **resultado en forma de gráfica** y en **diferentes niveles de agrupamiento** de datos.
    - **Agrupamiento** y **desagrupamiento** de **datos en forma interactiva**.
    - **Análisis** de **problemas** en términos de **dimensiones**.
        - Por ejemplo, permite analizar datos históricos a través de una dimensión tiempo.
    - **Control de calidad de datos** para asegura la **consistencia de la base** y la **relevancia** de los **datos** que contribuirán con la **toma de decisiones**.

Características

domingo, 4 de julio de 2021 05:32 p. m.

- **Está orientado a sujetos**
  - **NO se orienta a los procesos u operaciones** clásicas, como en el caso de los sistemas y diseños transaccionales. Sino que su **modelo operacional es orientado a los sujetos** u objetos de la organización.
    - Como en un sistema **OLAP** se **orienta** en **base** a **dimensiones** como pueden ser autos, facturas, etc.

- **Es integrado**
  - Los datos provenientes del ambiente transaccional u operacional se integran antes de ingresar en base a los sujetos u objetos antes mencionados para almacenarse en el Data Warehouse.

- **Es temático**
  - Desde el entorno operacional **solamente se añadirán los datos que se necesitan** en el proceso de **generación** de **conocimiento** del negocio.
  - Estos datos distribuidos por temas para facilitar la comprensión de los usuarios finales se reunen en una tabla de Data Warehouse.
  - Como toda la información se encuentra en un mismo lugar los requerimientos de información acerca de los clientes se responderán sin complicaciones.

- **Es variante en el tiempo**
  - Los **datos** en Data Warehouse **varían en el tiempo**. Esto significa que son rigurosos en un determinado momento y no en otro.
    - Una tendencia puede variar con nueva información, como por ejemplo un promedio de edad puede variar cuando se cargue nueva información sobre los clientes.

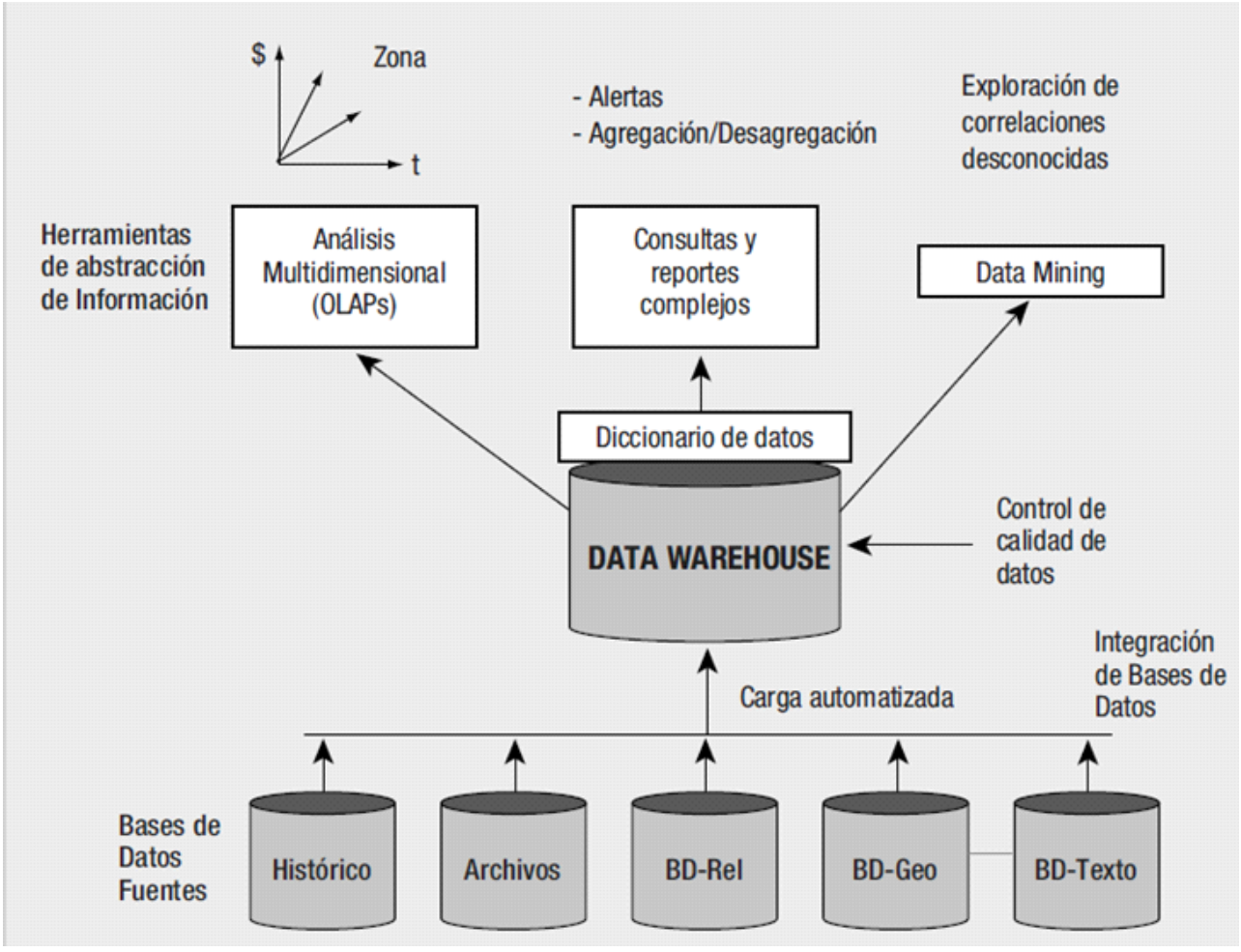
- **No es volátil**
  - El almacén de **información** de un Data Warehouse **se puede leer pero no admite ninguna modificación**.
  - En consecuencia, la **información es inalterable y sus actualizaciones no la cambian**.
  - **Sólo se incorporan las últimas variables.**

- **Es simple de manejar**
  - Las consultas son simples y rápidas, la información que se requiere consultar está previamente estructurada y precalculada para facilitar estas operaciones.
  - La creación y ejecución de estas consultas es sencilla y por lo tanto no requiere de un alto nivel de conocimiento técnico para el armado de consultas.

Arquitectura

domingo, 4 de julio de 2021 05:52 p. m.

- **Bases de Datos Fuentes**
  - **Distintos orígenes** (proceso de carga).
    - Por cada origen se construye un ETL (extract, transform and load) y se realiza el proceso de carga.
- **Herramientas de abstracción de la información**:
  - **Consultas y reportes complejos**: con **datos precalculados** normalmente a través de una herramienta
  - **Análisis multidimensional** (OLAPs): para **queries básicos.**
  - **Data mining**: búsqueda de **correlaciones desconocidas.**



Objetivos de un DW

domingo, 4 de julio de 2021 06:01 p. m.

- El **objetivo** de un ambiente de Data Warehouse consiste, principalmente, **en la conversión de los datos de las aplicaciones del ambiente transaccional (OLTP)** en datos integrados.
- Luego, es **necesario** que se los **almacene** en una **estructura que facilite el acceso de los usuarios finales** en un ambiente destinado a la **toma de decisiones (OLAP)**.
- Durante este proceso, la **totalidad** de los **datos se resumen y se incorporan al DW**, es decir, se los **transfiere de manera periódica**, de acuerdo con el análisis de negocios que se esté tratando.

Funcionalidades de un DW

domingo, 4 de julio de 2021 06:02 p. m.

- Las **funcionalidades** de **DW** se pueden subclasificar en **cinco grandes grupos**. Cada uno de ellos es **responsable** de un **conjunto de procesos específicos**, **indispensables** para el ambiente de soporte destinado a la **toma de decisiones**:

➢ Acceso a Fuentes (Source)

- **Procesos** que se aplican en las **bases de datos fuentes** a los **datos** que se **transferirán**.
- Algunos **procesos** asociados con la función de acceso a fuentes son por ejemplo, **mapeo, integración y muestreo de datos**.

➢ Carga (Load)

- La funcionalidad abarca diferentes procesos:
  - **Extracción**: es el primer paso de la preparación de los datos y comprende el **acceso a los datos** de los aplicativos.
    - Para la extracción existen **diferentes alternativas** que **equilibran** la **performance** y las **restricciones de tiempo** y de **almacenamiento**.
  - **Depuración**: es el proceso que verifica la **calidad** de los datos.
  - **Conversión**: es el último paso en la preparación de los datos que se cargarán en el DW. Este proceso necesita **reglas de conversión de valores** de aplicativos locales a globales e integrados.
  - **Carga de datos** es el proceso que **ingresa** los **datos** al DW.

- A cada **proceso** que **obtiene datos y los carga** en el **DW** desde una **fuente de datos distinta** se lo denomina **ETL**.
  - **ETL (extract, transform and load): extraer los datos, transformarlos y cargarlos.**

➢ Almacenamiento (Storage)

- El almacenamiento abarca la **arquitectura que se necesita para incluir varias vistas en DW**.
- Si bien se suele decir que Warehouse es un único almacén, **en realidad**, sus **datos** pueden estar **desperdigados en muchas bases** que **se manejan a través de diferentes DBMS's**.
- Tipos de **DBMS's**:
  - Los **relacionales** (RDBMS's)
  - Los **multidimensionales** (MDDBMS's).
    - MDDBMS, los **datos se organizan en un array de n dimensiones**. Cada una de ellas representa un aspecto del negocio que se analizará.

➢ Consultas (Query)

- El ambiente de consultas mediante sus herramientas OLAP **permite que el usuario dirija el análisis** y la **producción de reportes**.
  - **Data Mining**
    - Se encargan del análisis de los datos para verificar la existencia de correlaciones inesperadas entre ellos.
  - **Simulación de negocios**
    - Crean las herramientas necesarias para comprobar el impacto de las transformaciones en el ambiente de negocios y establecen, si se considera conveniente, nuevas reglas de organización que realimentarán los aplicativos operacionales.

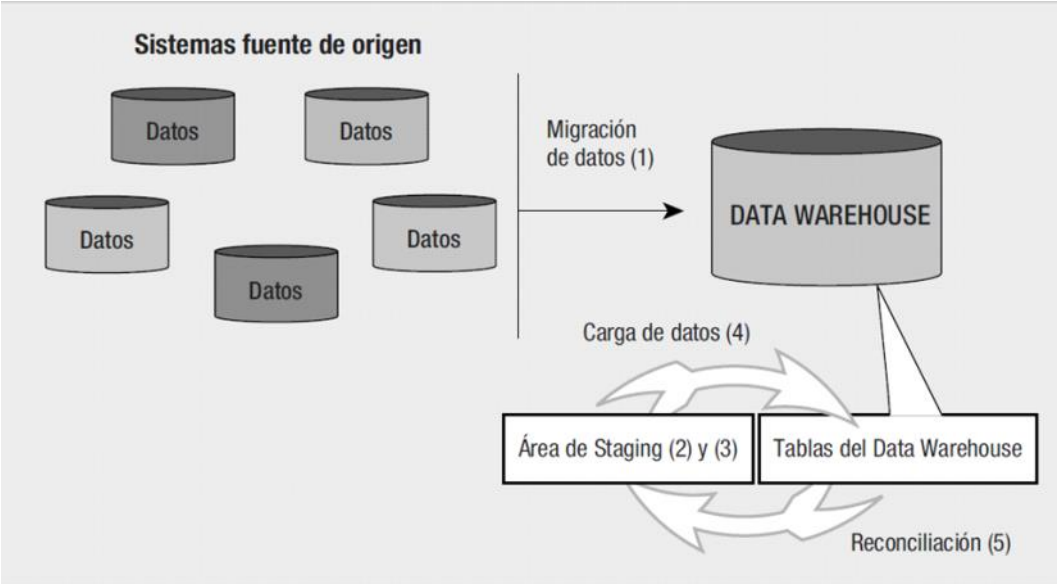
➢ Utilización de Metadatos (Meta Data)

- Un **metadato** es:
  - "toda aquella **información descriptiva** sobre el **contexto**, **calidad**, **condición** o **características de un recurso, dato u objeto** que tiene la finalidad de **facilitar su recuperación**, **autentificación**, **evaluación**, **preservación** y/o **interoperabilidad**"
- Los **metadatos** deben estar **disponibles** para el **análisis** que realizan los **usuarios**. En este caso, los **administradores** pueden **manejar** y **proveer** el **acceso** a través de los servicios de repositorio

Migración de datos

domingo, 4 de julio de 2021 06:50 p. m.

- La **migración** es **trasladar** los **datos desde** los sistemas seleccionados de **origen** hasta el **stage** de **DW** y luego una vez procesados (análisis, conversión, transformación o lo que fuese necesario) se **ingresan al DW** desde el **stage**.
    - **Sólo se moverán los datos solicitados por los usuarios** para la emisión de **reportes** o aquellos que se **utilizan** durante los **procesos** de **conversión** y **carga**, de esta manera se **previene** el **ingreso** de **información innecesaria**.
- Los **datos** que se moverán al **stage** de DW incluirán **datos referenciales** y **transaccionales**.
    - **Referenciales**: relacionados a la información de un sujeto. Como información de un cliente.
    - **Transaccionales**: relacionados a las operaciones. Como información sobre ventas de un cliente.



Depuración de datos

domingo, 4 de julio de 2021 06:59 p. m.

- La **depuración** de **datos** es **corregir para estandarizar el formato** y **completar cualquier valor requerido** por **DW**.
- **Este proceso contribuye** con la **identificación** de los **datos redundantes** que, durante el proceso de carga, no se ingresarán en DW.
- Para ello, se utilizan herramientas de software que migren, depuren y conviertan los datos. El retorno de la inversión justifica la compra de herramientas de software en vez de desarrollar scripts en SQL.
    - Los costos asociados a mantener y ampliar desarrollos propios de scripts SQL excederán significativamente al de comprar herramientas de software desarrolladas por terceros.

Ejemplo:

Tabla 5.1:

Nombre	Apellido	Empresa	Area	Telefono	Provincia
ANDRES	PEREZ	IBM	11	5355-2299	BA
Andres	Perez	I.B.M,	11	5355-2299	BA
Andrés	Pérez	IBM SA	11	5355-2299	
Andes	Peres	I.B.M. S.A.	11	5355-2299	BA
Martín	López	Bunge S.A.	351	272-2700	CO

Luego del proceso de depuración la tabla resulta de la siguiente manera:

Tabla 5.2:

Nombre	Apellido	Empresa	Area	Telefono	Provincia
Andrés	Pérez	I.B.M. S.A.	11	5355-2299	BA
Andrés	Pérez	I.B.M. S.A.	11	5355-2299	BA
Andrés	Pérez	I.B.M. S.A.	11	5355-2299	BA
Andrés	Pérez	I.B.M. S.A.	11	5355-2299	BA
Martín	López	Bunge S.A.	351	272-2700	CO

Conversión de datos

domingo, 4 de julio de 2021 07:03 p. m.

- El objetivo de la **conversión** de los **datos** es **cambiar los datos con el formato** y la **estructura requeridos** por el **DW**.
- En el desarrollo de las **reglas de conversión** para este proceso, sólo se **utilizarán** aquellos elementos de **datos que se requieran** para **DW**. Si existieran **otros que resultaran innecesarios**, se **prevendrá su ingreso** en DW y no se los incorporará en las sentencias de conversión o carga.

Carga de datos

domingo, 4 de julio de 2021 07:07 p. m.

- La **renovación completa**
    - Comienza **truncando las tabla**s en Data Warehouse y luego **cargándolas** con **todos los datos requeridos**. Esta alternativa **puede prevenir que datos no deseados** ingresen a Data Warehouse abarcando **condiciones** en las **sentencias de carga**

- La **renovación incremental**
    - **Identifica** los **cambios que se produjeron en los datos origen desde la última vez que se cargó** Data Warehouse y **luego inserta, actualiza o borra registros** de datos en cada tabla de Data Warehouse como se lo solicite.

Conciliación de datos

domingo, 4 de julio de 2021 07:09 p. m.

- El proceso de **conciliación valida** la que la **información** haya sido **cargada correctamente** en el **DW**. Este proceso identifica los problemas de datos que, si no se les diera importancia, pasarían los controles de prevención.
- Este proceso se diseña para proveer **veracidad** e **integridad**, también para la **identificación** de los **datos que no concuerdan** con la **información** que contiene el sistema de **origen**.
- Para ello se debe v**erificar** tanto su **cantidad** (que se corresponda con los datos de entrada) como su **calidad** (que no haya nulos, etc.) con respecto a la fuente:
  - **La calidad de datos:** la exactitud se evalúa con el uso de totales de control sobre los elementos de datos seleccionados, que luego se compararán con los resultados anticipados.
  - **La cantidad de datos:** la integridad se determina **cuantificando** el **número de registros** y **comparando** los resultados con el **número de registros anticipados**.

- ***C*onciliación completa**
  - Al **finalizar** cada proceso de **carga**, se realiza una **conciliación completa** que **compara** la **información** de **Data Warehouse** con la del **sistema origen** correspondiente.
- **Conciliación por Fase**
  - La **conciliación** se realiza **después de cada etapa del flujo del proceso de datos**, cuando no es factible una conciliación completa porque debido al número de sistemas de origen o a la complejidad de los procesos de depuración o conversión.
  - Con la **conciliación por fase**, se determinan la veracidad e integridad de los datos luego de cada una de las siguientes etapas:
    - **Migración de datos:** después de que los datos del sistema origen han sido migrados al *stage* deL DW, se realiza la conciliación entre los datos del sistema origen y los del *stage* deL DW.
    - **Depuración:** cuando termina el proceso de depuración, se realiza la conciliación entre los datos no depurados, el listado de excepciones y los datos depurados del *stage* del DW.
    - **Conversión:** una vez que finaliza el proceso de conversión, se produce la conciliación entre los datos depurados, la lista de excepciones y los datos convertidos del *stage* del DW.
    - **Carga:** después de terminado el proceso de carga, se hace la conciliación entre los datos convertidos del *stage* del DW.

Data Marts (DM)

domingo, 4 de julio de 2021 07:22 p. m.

- Se denomina **Data Marts** son pequeños Data Warehouse **acotados en volumen a una temática particular** o **funcionalidad de la organización**, estos representan **vistas multidimensionales** de **cada área**. **Información segmentada** de un **sector** de la organización.
- **Optimizan** la **distribución de información** útil para la **toma de decisiones** y se enfocan al **manejo de datos resumidos** o **de muestras**, más que a la historia presentada con detalle.
- Los **Datas Marts** deben su popularidad a que **disminuyen** de manera significativa los **costos asociados a su creación y operación**.

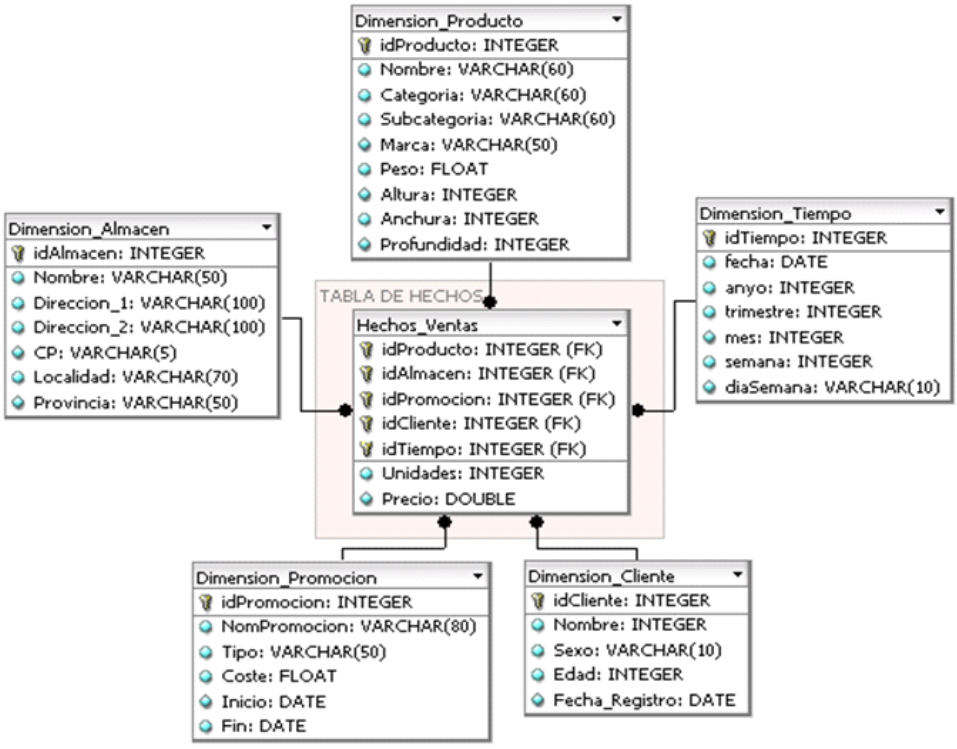
---

domingo, 4 de julio de 2021 07:26 p. m.

Implementación de DW

domingo, 4 de julio de 2021 07:39 p. m.

- Un **DW** o un **Data Marts** puede ser implementado en **DBMS's Multidimensionales** o **Relacionales**.
- Para implementar un DW en un RDBMS se utiliza lo que se denomina el **Modelo Estrella (STAR MODEL)**
- **Modelo Estrella*:*** desde información dimensional **crea un modelo de tablas relacionales para trabajarlas como si fuesen dimensiones**.



- ***Tabla de Hechos (Fact Table)***
  - Registra **medidas** o **métricas** de un **evento específico (hecho)**, generalmente consisten de valores numéricos (datos asociados específicamente con el evento) y **claves foráneas** (PK de las dimensiones) que **referencian** a **tablas de datos dimensionale**s que guardan **información descriptiva** del hecho.
  - Se diseñan para **contener detalles uniformes a bajo nivel** (referidos como "granularidad" o "grano"), o sea que los hechos pueden registrar eventos a un gran nivel de atomicidad.
- ***Tabla de Dimensiones (Dimension Table)***
  - Las **dimensiones** pueden definir una amplia variedad de características.
  - Las tablas de **dimensiones** generalmente tienen un **bajo número de registros**, en comparación a las tablas de hechos, pero **cada registro puede tener un gran número de atributos** para **describir** los datos del **hecho**.

Data Mining

domingo, 4 de julio de 2021 07:57 p. m.

- **Conjunto** de **técnicas** que se utilizan para la **obtención** de la **información implícita** en grandes bases de datos.
- Se encarga, a **través** de un **conjunto de herramientas y técnicas algorítmicas**, de **buscar** los **patrones de interés ocultos**, que son los que permiten la **anticipación** de **futuros acontecimientos** gracias a la **predicción de acontecimientos** o al **pronóstico** de **situación** con **cierto grado de probabilidad.**
- Estas herramientas **"explota"** la **información** en busca de **patrones ocultos**, encontrando **información predecible** que **un experto no puede llegar a encontrar.**
    - El término **"explotar"** refiere a que de un bloque de mucha información se va explotando para cruzar todos los datos entre si hasta lo más atómico posible para compararlos.
- Se **utiliza** para poder **decidir quién es el público objetivo del "producto"** en cuestión (tu mercado), por ejemplo para saber a qué horizonte dirigir una campaña de marketing.

Características

lunes, 5 de julio de 2021 12:28 p. m.

- Los procesos de Data Mining corren sobre bases de datos de gran volumen; esto se produce por dos aspectos fundamentales que se analizarán a continuación:
  - **Gran cantidad de columnas**
    - **Cuantas más columnas** se especifiquen en DW, **mayor** será el **nivel de análisis y de detalle en Data Mining**, dado que realiza **diferentes combinaciones** entre los **patrones especificados** —en este caso, las columnas predefinidas—.
    - La **cantidad** de **conclusiones** que entregue estará en **estrecha relación** con el **nivel de combinación** que realice.
  - **Gran cantidad de filas**
    - **Para disminuir la cantidad de errores de estimación y desvíos** se necesita que las **tablas tengan** la **mayor cantidad de filas** posibles que provean toda la información histórica disponible.

- Para que Data Mining se pueda ejecutar y cumplir con su objetivo, debe tener las siguientes características:
  - **Recolección de datos en gran escala**
    - **Unifica** el contenido de la **información** de **todas** las **bases de datos disponibles**, internas o externas.
    - Como ya se mencionó, **se disminuyen los errores y desvíos si la información disponible contiene amplitud y profundidad** porque, de este modo, mayor será la aproximación o proyección que se obtenga de la tecnología.
  - **Alta Tecnología y gran almacenamiento**
    - Como Data Mining procesa un volumen de información considerable y realiza un importante número de comparaciones / combinaciones, necesita múltiples y veloces procesadores, una gran capacidad de memoria RAM y una gran memoria secundaria debido a los procesos intermedios de recolección y combinación de datos e información que ejecuta esta tecnología.
  - **Algoritmos de Data Mining:**
    - Data Mining funciona con la **aplicación** de **diversas herramientas algorítmicas** que son las que permiten la **búsqueda** de **información oculta**.

- La tecnología de Data Mining, con bases de datos de suficiente tamaño y calidad, **genera nuevas**

oportunidades de negocios que proveen las siguientes capacidades:

- **Predicción automatizada de tendencias y comportamientos**
    - Al automatizar la búsqueda de información predecible en grandes bases de datos, **puede inferir**, **ante** una **nueva situación** o **estímulo** determinado, cuál sería el **comportamiento futuro.**

- **Obtención automatizada de modelos previamente desconocidos**
    - Es necesario la utilización de DM para que barra de un solo paso, a través de sus herramientas algorítmicas de búsqueda de información oculta, el DW e **identifique los patrones de tendencia desconocidos por la organización**.
    - Para descubrirlo evalúa **todos** los **parámetros** que conforman el **comportamiento** de los **actores**, y los combina con el fin de obtener una heurística de comportamiento que **identifique**, con una **determinada probabilidad**, el **interés** de los **clientes** por ese **producto**.

Herramientas Algorítmicas

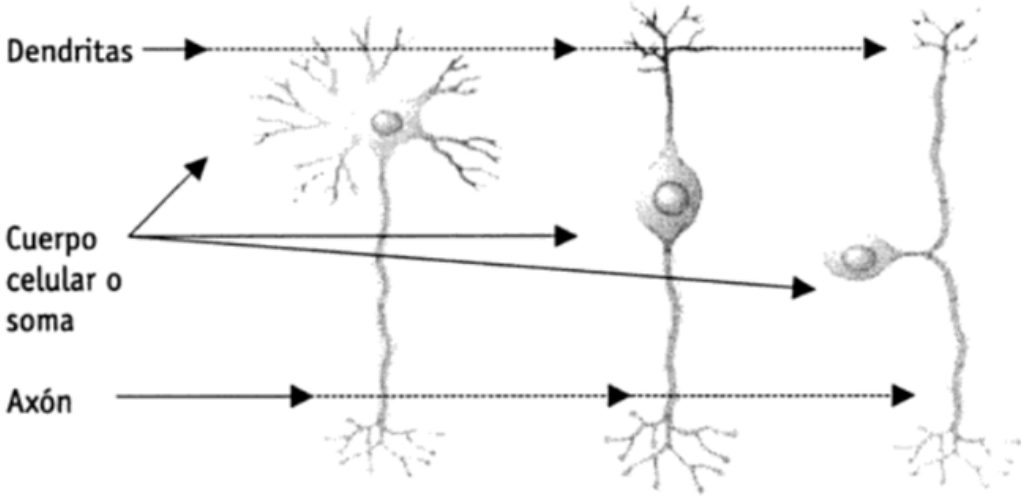
lunes, 5 de julio de 2021 12:55 p. m.

- Dentro de las técnicas más utilizadas en Data Mining se encuentran:
  - **Redes neuronales artificiales**
    - Son modelos predecibles, de características no lineales que **aprenden a través del entrenamiento** y semejan la **estructura** de una **red neuronal biológica**.
  - **Algoritmos genéticos**
    - Son técnicas de optimización con un **diseño** basado en el **concepto de evolución** y que utilizan **procesos** como las **combinaciones genéticas, las mutaciones y la selección natural**.
  - **Arboles de decisión**
    - Estructuras cuya forma representa la copa de un árbol y que **representan conjuntos de decisiones**.
    - **Estas decisiones** son las que **generan** las **reglas que clasifican un conjunto de datos**, que se segmentan mediante búsquedas arboladas.
    - Dentro de los métodos específicos de árboles de decisión se incluyen, también, los Árboles de Clasificación y Regresión.

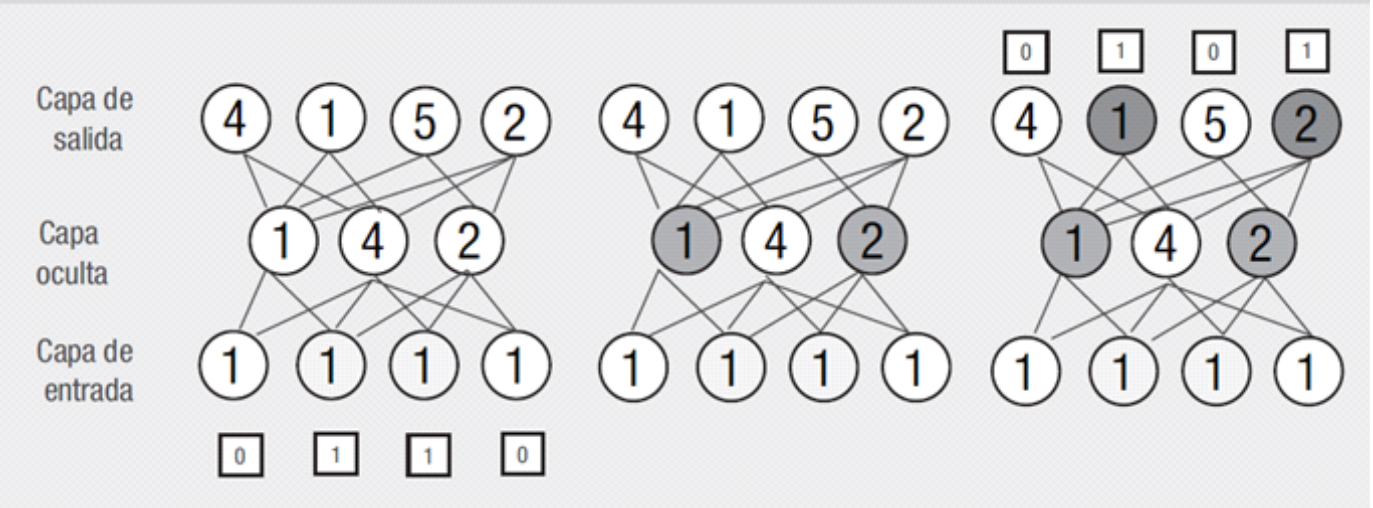
Red Neuronal

lunes, 5 de julio de 2021 01:02 p. m.

- La red neuronal artificial es un **método de resolución de problemas** que **emula** el modo de **conexión** de las **neuronas del cerebro**.
- Esta red posee **capas de unidades procesadoras** (nodos) que se unen por conexiones direccionales.



- **Si la suma** de **todas** las **entradas** que **ingresan** en una de estas neuronas virtuales es **mayor** que el **umbral de activación de la neurona**, **esta se activa y transmite su propia señal** a las de la **siguiente capa**. Por lo tanto, el **patrón de activación se propaga hasta** que llega a la **capa de salida** que lo devuelve como **solución a la entrada presentada**.
- Entrenamiento de la red neuronal
    - De la misma manera que en el sistema nervioso de los organismos biológicos, **con el transcurso del tiempo, una red neuronal aprende y afina su rendimiento** gracias a la **repetición de rondas** en las que **ajusta sus umbrales hasta que**, para cualquier entrada, la **salida real coincide con la deseada**.



- Aquí vemos una sencilla red neuronal con tres capas: la de entrada, con cuatro neuronas; la oculta, con tres y, por último, la de salida, con cuatro.
- El diagrama muestra cómo la red neuronal recibe una cadena de entrada y cómo la activación se extiende por la red hasta producir una salida.
- El **umbral de activación** de **cada neurona** se representa por su **número**. **Para que se excite debe recibir, por lo menos, esa cantidad de entradas**.

- Existen algoritmos de optimización para el entrenamiento de las redes neuronales:
  - **Ascenso a colina** o **voraces**
    - Comienza con la solución de un problema que tiene a mano que, normalmente, se elige al azar. Después, la cadena se muta y, si ésta proporciona una solución con mayor amplitud que la anterior, se conserva la nueva; en caso contrario, la actual.
    - Este algoritmo **se repite hasta que no se pueda encontrar una mutación que provoque un incremento en la aptitud de la solución actual**, y ésta se devuelve como **resultado**.
    - Esta técnica se denomina ascenso a colina porque, en general, se representa con un paisaje en el que se encuentran todas las soluciones posibles de un problema particular.
      - Cada solución, a la vez, se constituye por un conjunto de coordenadas de ese paisaje.
      - La mejores soluciones están a mayor altitud y forman colinas y picos; las peores, a menor altitud y forman valles.
      - Un "trepacolinas" es, en consecuencia, un algoritmo que se inicia en un punto de paisaje y se mueve hacia arriba de la colina.
    - Este tipo de algoritmo también se lo denomina **"voraz"** porque **siempre hace la mejor elección en cada paso, con la esperanza de que se obtendrá el mejor resultado global**.

  - **Recocido simulado**
    - En el recocido simulado una función de aptitud **define una solución candidata**. Esta clase de recorrido añade, además, el concepto de "temperatura", que es una cantidad numérica global que disminuye de manera gradual. **En cada uno de sus pasos, esta solución muta**.
    - **La aptitud de la nueva solución se compara con la anterior** y, **si es mayor**, se la **conserva**. **Si ocurre lo contrario**, el algoritmo **decide si la conserva o la descarta sobre la base de la temperatura**.
      - Si ésta es alta, se conservan incluso los cambios que causan disminuciones significativas en la aptitud y se utilizan para la siguiente ronda.

        - Sin embargo, a medida que disminuye la temperatura, el algoritmo tiende a aceptar sólo los cambios que aumentan la aptitud.

    - Finalmente, **cuando la temperatura alcanza el cero y el sistema se "congela", la configuración que exista en ese punto se convierte en la solución**.

Algoritmos Genéticos

lunes, 5 de julio de 2021 01:24 p. m.

- **Son algoritmos de optimización**, o sea, **tratan de encontrar la mejor solución a un problema dado entre un conjunto de soluciones posibles**.
    - Los mecanismos que utilizan los AG para llevar a cabo esa búsqueda consisten en **procesos que se asemejan a la evolución biológica**, de allí el nombre de algoritmos genéticos.
- Trabaja sobre el concepto de la ***mutación***
    - Dada una **población de soluciones**, y en base al **valor de la función objetivo para cada uno de los individuos (soluciones)** de esa población, **se seleccionan los mejores individuos** (que son aquellos que minimizan la función objetivo) **y se combinan para generar otros nuevos**.
        - Este proceso se repite cíclicamente hasta probar todas las combinaciones y encontrar la óptima.
- Es simillas con el proceso que se da en la naturaleza, en el que los individuos compiten por su supervivencia.
    - Los mejor adaptados al medio (los que pueden optimizar la función objetivo) sobreviven y transmiten su material genético a las futuras generaciones.

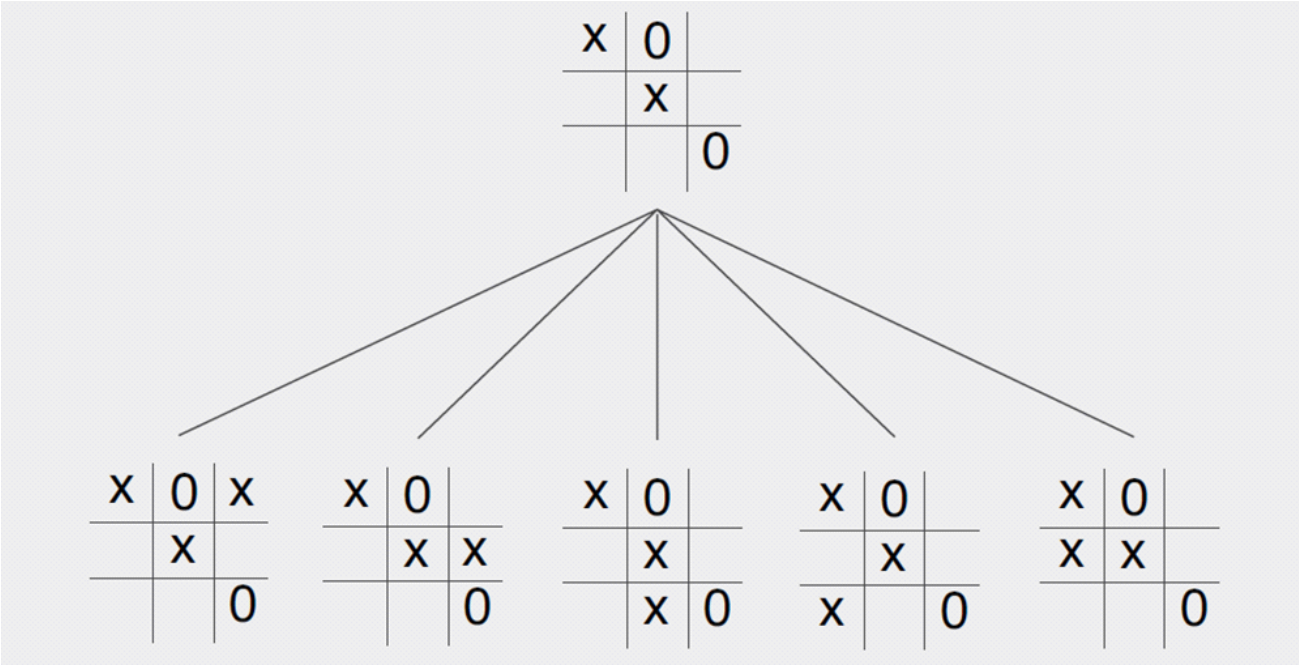
Pasos de los Algoritmos Genéticos

1. **Definir la solución**
2. **Filtrar**
    a. **Aplicarle la función objetivo.**
    b. **Ordenar** los **individuos** en función de los valores obtenidos.
    c. **Seleccionar los mejores individuos** (soluciones) para el cruce.
3. **Cruzar** los individuos (mezclar sus características).
4. **Mutación** de los descendientes.
5. **Inserción**.
6. **Si se cumple** la función objetivo **”terminar”**
    a. De lo contrario volver al paso 2.

Arboles de decisión

lunes, 5 de julio de 2021 01:35 p. m.

- Son una técnica de programación que **permite analizar decisiones secuenciales basadas** en el uso de **resultados y probabilidades** asociadas (heurística de ocurrencia, o sea probabilidad de ocurrencia).
- Los árboles de decisión, se utilizan en la Inteligencia Artificial, especialmente en los denominados **sistemas expertos**, que se basan en grandes bases de datos, en las que se cargan reglas de decisión que encuentran su fundamento en la experiencia de los expertos en una ciencia determinada sobre la que versará el sistema. De esta manera, se lo puede utilizar para establecer un diagnóstico determinado, en el que se evalúa todos los caminos posibles dentro del árbol.



- **Ventajas**
  - **Resume los ejemplos de partida** y permite la **clasificación de nuevos casos** siempre y cuando no existan modificaciones sustanciales en las condiciones que generaron los ejemplos que sirvieron para su construcción.
  - **Facilita la interpretación de la decisión adoptada** ya que permite **regenerar** el **camino decisorio** aplicado.
  - Proporciona un **alto grado de comprensión del conocimiento** utilizado en la **toma de decisiones**.
  - Explica el comportamiento respecto a una determinada tarea de decisión.
  - Reduce el número de variables independientes.
  - Es una magnífica herramienta para el control de la gestión empresarial.

Ventajas de DM

lunes, 5 de julio de 2021 03:15 p. m.

- **Contribuye** con la **toma de decisiones estratégicas** y **proporciona** un **sentido automatizado** para **identificar información clave** desde volúmenes de datos generados por procesos tradicionales y de *Business Intelligence*.
- Permite a los usuarios **dar prioridad a decisiones y acciones** e **indica** los **factores** que tienen una **mayor incidencia**, qué **segmentos** de clientes son **desechables** y qué **unidades de negocio** son **sobrepasadas** y por qué.
- **Genera modelos descriptivos**
    - En un contexto de objetivos definidos en los negocios, permite a las organizaciones, sin que se considere la industria o el tamaño, **explorar automáticamente, visualizar y comprender los datos e identificar patrones, relaciones y dependencias que impactan en los resultados finales**.
- **Genera modelos predictivos**
    - Permite que **relaciones no descubiertas e identificadas** a través del proceso de Data Mining **se expresen como reglas de negocio o modelos predictivos**.

domingo, 11 de julio de 2021 11:46 a. m.

Unidad 9 - Estructura DBMS

Conceptos

domingo, 4 de julio de 2021 08:11 p. m.

- ***BD***
    - Conjunto de datos interrelacionados que se ajustan a una serie de modelos preestablecidos que recogen información de interés de objetos del mundo real.
- ***DBMS (Data Base Management System)***
    - **Software para la administración de la base de datos**.
    - Su misión es **proporcionar mecanismos de acceso a los datos para almacenar, definir y recuperar información** de forma eficiente.
    - Existen además una serie de aplicaciones en torno al DBMS que aportan interfaces sencillos para manejar los datos.

Propiedades de DBMS

lunes, 5 de julio de 2021 03:32 p. m.

Para que un producto se considere un **Motor de Base de Datos (DBMS)** debe cumplir con determinadas propiedades.

Dichas propiedades se agrupan dentro del acrónimo ***ACID***.

- ***ACID***
  - **A**tomicity (**Atomicidad**)
  - **C**onsistency (**Consistencia**)
  - **I**solation (**Aislamiento**)
  - **D**urability (**Durabilidad**)

- ***Atomicidad***
  - Propiedad que **asegura que una operación se ha realizado o no**, y por lo tanto **ante un fallo del sistema no puede quedar a medias**.
  - Se dice que una **operación es atómica** cuando **es imposible** para otra parte de un sistema **encontrar pasos intermedios**.
  - **Si esta operación consiste en una serie de pasos, todos ellos ocurren o ninguno**.
    - Por ejemplo, en el caso de una transacción bancaria o se ejecuta tanto el depósito y la deducción o ninguna acción es realizada. Es una característica de los sistemas transaccionales.

- ***Consistencia* (*Integridad*)**
  - La propiedad de **consistencia** sostiene que **cualquier transacción llevará a la base de datos desde un estado válido a otro también válido**.
  - Se **ejecutan** aquellas **operaciones** que **no van a romper las reglas y directrices de *Integridad*** de la base de datos.
  - El concepto de **consistencia** está **fuertemente relacionado** con la propiedad de **atomicidad** porque **siempre se controla** que **toda operación se realice completamente o que no se realice manteniéndose en un estado consistente la base de datos**.
  - *"La Integridad de la Base de Datos nos permite asegurar que los datos son exactos y consistentes, es decir que estén siempre intactos, sean siempre los*

esperados y que de ninguna manera cambian ni se deformen. De esta manera podemos garantizar que la información que se presenta al usuario será siempre la misma."

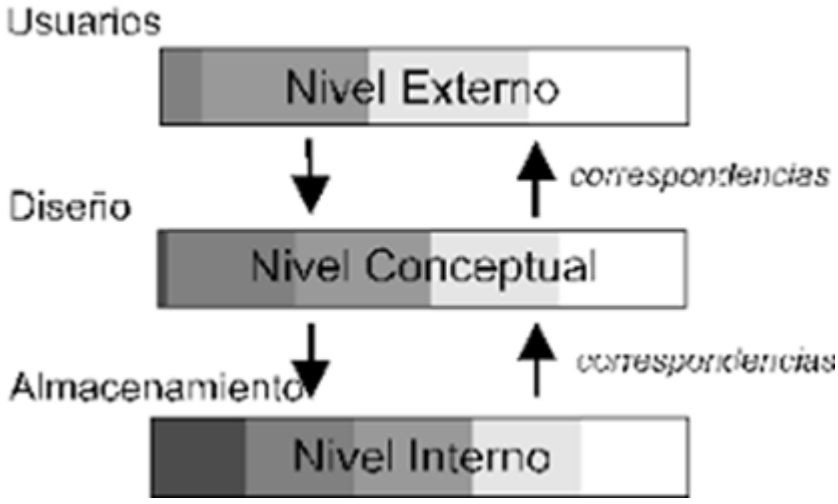
- ***Aislamiento (Isolation)***
  - Posibilidad de **aislar operaciones concurrentes** a la **misma información**.
  - Propiedad que asegura que **una operación no puede afectar a otras**.
  - Esto asegura que **la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error**.
  - Esta propiedad define **cómo y cuándo** los **cambios producidos** por una **operación** se hacen **visibles** para las **demás operaciones concurrentes**.
  - El **aislamiento** puede alcanzarse en **distintos niveles**, siendo el **parámetro esencial** a la hora de seleccionar un DBMS.
  - El concepto de **aislamiento** se relaciona con la **consistencia** y **atomicidad** de un DBMS ya que para **poder aislar operaciones concurrentes** a la misma información es **necesaria la atomicidad** de operaciones, y **gracias a esto** la **base de datos siempre** queda en un **estado consistente**.

- ***Durabilidad* (*Persistencia*)**
  - La **Persistencia** es **preservar** la **información** de **forma permanente** (guardar), pero **a su vez** también se refiere a **poder recuperar la información** (leer) para que pueda ser nuevamente utilizada.
  - Esta propiedad asegura que una **vez realizada la operación, esta persistirá y no se podrá deshacer** aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera.

Arquitectura ANSI de un DBMS

lunes, 5 de julio de 2021 03:41 p. m.

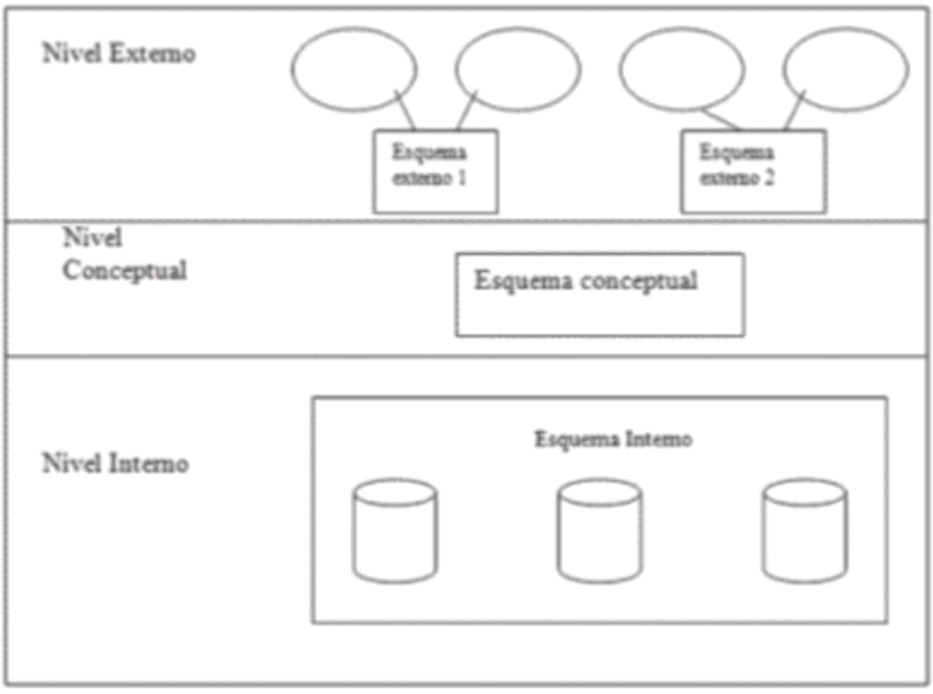
- La **Arquitectura de un DBMS** se compone de **tres niveles** que hoy en la arquitectura de software son conocidas como **capas**.
- Esta arquitectura si bien viene desde la década del 70 es la utilizada actualmente para la mayoría de desarrollo en tres capas.
- El DBMS está conformado por los siguientes **niveles**:
  - **Externo o de usuario**
  - **Conceptual o lógico**
  - **Interno o físico**



<u>Cada capa tendrá una funcionalidad:</u>

- **Externo**
  - Atender el usuario, capa de presentación del sistema.
  - **SSMS** (**Management Studio**) / Oracle, interfaz del sistema con la que interactúa el usuario.
- **Conceptual**
  - **Reglas lógicas** del motor: que se puede y que no hacer.
  - En ella se encuentra el analizador sintáctico (parser) y semántico (scanner).
  - **Developer** genera **modelos lógicos** para desarrollar aplicaciones.
    - Ejemplo: qué puedo hacer en un store procedure.
- **Interno**
  - Mantener la durabilidad y persistencia de la información.
  - **Almacenamiento**, cómo persistir los datos.

ARQUITECTURA DE LOS DBMS



Estas 3 capas le permiten a los usuarios:

- **Permite vistas de usuario independientes y personalizadas**
    - Cada usuario debe ser capaz de acceder a los datos, pero tiene una vista personalizada diferente de los datos.
    - Éstos deben ser independientes: los cambios en una vista no deben afectar a las demás.

- **Oculta los detalles físicos de almacenamiento a los usuarios**
    - Los usuarios no deberían tener que lidiar con los detalles de almacenamiento de la base de datos.

- **El administrador de la base de datos debe ser capaz de cambiar las estructuras**
    - Modifica la estructura de almacenamiento de la BD sin afectar la vista de los usuarios.

- **La estructura interna de la base de datos no debería verse afectada por cambios en los aspectos físicos del almacenamiento**
    - Por ejemplo, un cambio a un nuevo disco.

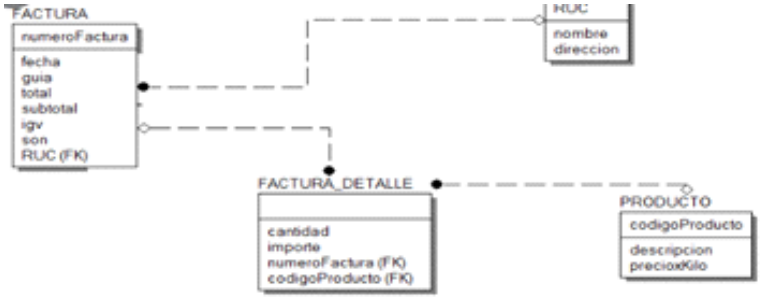
Visto desde el Almacenamiento

lunes, 5 de julio de 2021 03:45 p. m.

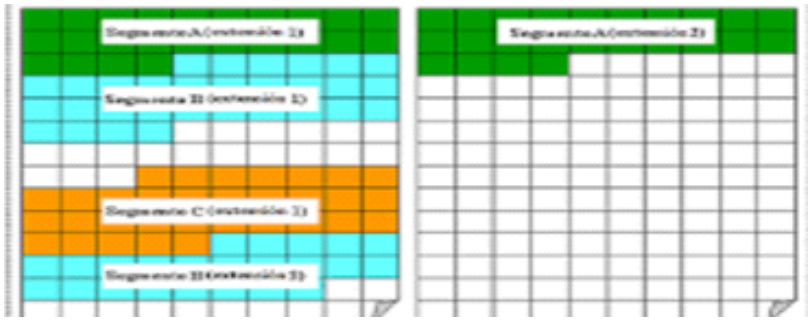
- **Nivel externo (Vistas individuales de los usuarios)**
  - Una vista de usuario describe una **parte** de la base de datos que es **relevante para un usuario en particular**.
  - **Excluye datos irrelevantes**, así como los **datos** que **el usuario no está autorizado a acceder**.

Alimentación	100	367	369	371	373	210	377
Viáticos	370	377	389	401	413	240	397
Estudios	375	387	409	431	453	270	417
Ocio	380	397	429	461	493	300	437
Hogar	385	407	449	491	533	330	457
Mascotas	390	417	469	521	573	360	477
Servicios	395	427	489	551	613	390	497
Personal	400	437	509	581	653	420	517
Celular	405	447	529	611	693	450	537
Otros	410	457	549	641	733	480	557
TOTAL							

- **Nivel Conceptual (Vista conceptual)**
  - El nivel **conceptual** es una forma de **describir los datos que se almacenan** dentro de la base de datos y **cómo los datos están relacionados entre sí**.
  - Permite visualizar el **diseño lógico** de los **datos almacenados**.
    - Estructurado como DER, **tiene que conocer objetos de la DB y sus atributos** para hacer los **análisis.**
  - Este nivel **NO** especifica cómo se almacenan físicamente los datos.



- **Nivel interno (Vista de almacenamiento)**
  - El nivel interno implica la **forma en que** la base de datos **se representa físicamente en el sistema informático**.
  - Describe **cómo los datos se almacenan en la base de datos** y en el **hardware del equipo**.
  - Los datos se almacenan de **forma secuencial sin un orden lógico**.



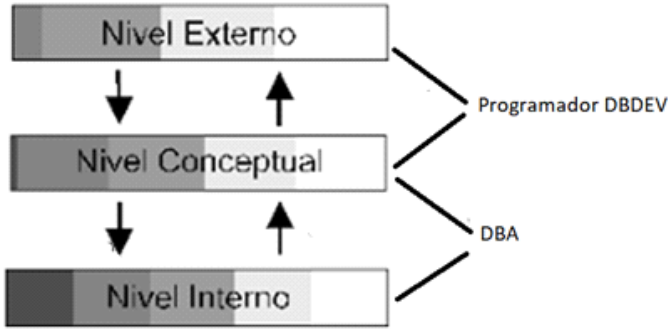
Visto desde la Funcionalidad

lunes, 5 de julio de 2021 03:48 p. m.

- **Nivel Externo**
  - Es el nivel en el que **el usuario interactúa con el DBMS**, entendiendo que el usuario puede ser un usuario final o alguna aplicación o lenguaje de programación.
  - El Developer **desarrolla aplicaciones** desde PL_SQL.
  - Utilización de una herramienta por parte del usuario.
  - El **DBA (Administrador)** puede **utilizar herramientas** para **administrar** la **base de datos**

- **Nivel Conceptual**
  - Es el **nivel** donde **radica la lógica del DBMS**, o sea, **donde se definen las reglas de lo que se puede y no se puede hacer**.
  - En esta capa se encuentra el analizados sintáctico (parser) y semántico (scanner).
  - El Developer genera modelos lógicos para desarrollar aplicaciones.
  - El DBA define el mejor nivel de diseño desde el punto de vista del DBMS.

- **Nivel interno**
  - Es el **nivel** que **maneja la persistencia de la información en el DBMS**, o sea, **como se almacena la información**.
  - El **DBA define la forma en que se van a almacenar los datos y que se puede hacer o no a nivel programación** en función de la configuración del DBMS.



Componentes de un DBMS

lunes, 5 de julio de 2021 03:51 p. m.

- **IPL (Initial Program Loader)**
    - Es un **programa de carga inicial** que **permite levantar el servicio del DBMS** y **disponer** la **estructura de memoria, cache y disco** para el **procesamiento** de las **operaciones**.

- **User Manager**
    - Es el módulo **encargado de manejar los perfiles, usuarios y roles de acceso** al **DBMS**.
    - Es el encargado de **manejar todos** los **componentes relacionados** con la **seguridad** del **DBMS**.
        - Esto incluye:
        - **Administración de Usuarios**
        - **Permisos de acceso**
        - **Seguridad Vertical**
            - Ej: Si el usuario **puede o no acceder** a una tabla
        - **Seguridad Horizontal**
            - Ej: Poder **acceder a una tabla pero solo a los campos que corresponden a un usuario**.

- **File Manager**
    - Es el módulo **encargado** de la **administración *lógica* de los archivos** que componen al **DBMS**.
    - Al igual que un Sistema Operativo maneja su propia **FAT (File Allocation Table), es interna del DBMS**.
        - Los datos del DBMS están encapsulados, o sea **desde el sistema operativo NO puedo acceder a estos** sino que si o si debo acceder a través del **DBMS**.
    - *"Una Base de Datos es un Sistema Operativo pero sin manejo de periféricos"*
        - Mismas funcionalidades que un SO pero no maneja drivers de periféricos, esto significa que no puede mostrar algo en pantalla, no puede imprimir, no puede acceder directamente a disco para leer / escribir sino que utiliza el SO para realizar estas funciones.
    - Dentro de sus funciones se encuentran:
        - **Creación de archivos**
        - **Eliminación de archivos**
        - **Acceso a los archivos**
        - **Interconexión con el User Manager para el acceso**

- **Disk Manager**
    - Es el módulo **encargado de la administración *física* de la información persistida** en el **DBMS**, **en su propio disco** (espacio brindado por el SO al DBMS).
    - Dentro de sus funciones se encuentran:
        - **Asignación de espacio de almacenamiento**
        - **Eliminación de espacio liberado**
        - **Acceso a la información física**
        - **Comunicación con el SO para el acceso al Disco**

Flujo entre Componentes

domingo, 11 de julio de 2021 08:08 p. m.

Ejemplo de flujo

Si se quisiera hace un SELECT * FROM Clientes WHERE id = 7 entonces

1. **Usuario** se conecta al **DBMS** por el administrador corporativo
2. **DBMS** valida acceso con el **User Manager.**
3. **User Manager** se comunica **con File Manager** para **generar sesión del usuario.**
4. **Usuario ingresa** la **consulta** a realizar, **la recibe** la **capa** de Usuario (**Externa**)
5. La **capa externa le pasa la consulta a la capa Conceptual** para **analizar** si la consulta es **correcta sintáctica y semánticamente**.
6. Como se necesita consultar la tabla Clientes **se le pide al File Manager** que **busque en la FAT la página** en **donde** se hallan **almacenados** todos los registros de **clientes**.
7. El **File Manager** le **indica** al **Disk Manager** que le **devuelva la información de la página** deseada, ya seleccionando por el id deseado.
8. La **información requerida burbujea** a través de las capas **hasta la capa externa** para **dar respuesta** a la **consulta** realizada por el **usuario**.

Nivel Interno de Almacenamiento

lunes, 5 de julio de 2021 03:54 p. m.

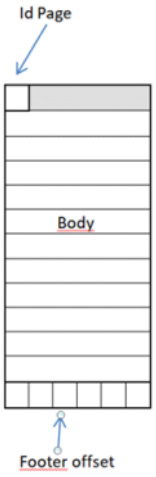
Existen **dos** técnicas de **administración** de **memoria:**

- **Segmentación**
  - Divide la memoria en **segmentos**, **cada uno** de los cuales **tiene** una **longitud variable**.
- **Paginación**
  - Divide la memoria en **páginas**, **cada una** de las cuales **es de longitud fija y de la misma longitud**.
  - **Este es el utilizado en los DBMS, ya que heredó la estructura de los mainframe.**

Paginación

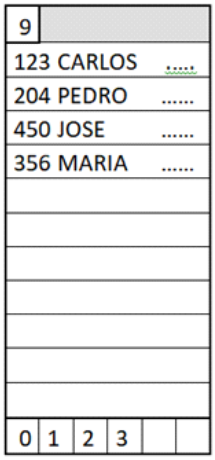
lunes, 5 de julio de 2021 03:57 p. m.

- **Página:** Una página tiene tres componentes el ***id page***, el ***body page*** y el ***footer offset***.
  - **Id**
    - **Es la identificación de la página, las cuales se encuentras numeradas y contiguas.**
  - **Body**
    - Es el cuerpo de la página donde se almacena la información.
    - Este cuerpo está dividido en unidades como registros equivalentes a renglones de una hoja.
  - **Footer offset**
    - Es el pie de página tiene tantas entradas como registros (renglones) que contenga la página. Actúa como el "índice" de la página.



Los datos se almacenan en ***body page*** de la página en función como van ingresando, y en el ***footer offset*** se registra la posición relativa de la fila que se encuentra en esa posición.

Es un **acceso directo a la información**.



Pos	Id	Nombre
0	123	CARLOS
1	204	PEDRO
2	450	JOSE
3	356	MARIA

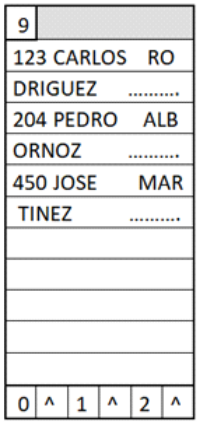
Fragmentación

domingo, 11 de julio de 2021 08:11 p. m.

En función del tamaño de la fila, puede producirse que sobre espacio en el almacenamiento lo que cual generará una fragmentación.

- ***Fragmentación Externa***
  - Se produce cuando el **tamaño de página es mayor al tamaño del cluster** mediante el cual asigna espacio el disco, entonces la página se almacena fragmentada en dos clústeres.
    - Cluster: Unidad más pequeña de almacenamiento de un SO.
  - Cuando el **tamaño de página es menor al tamaño del clúster la página no se almacena fragmentada** pero cada lectura de clúster contendrá mucho dato basura (información innecesaria). Cada lectura tardara más de lo debido.
  - **Cuanto más parecidos sean el tamaño de clúster y el tamaño de las paginas se producirá menos fragmentación externa y quedara menos dato basura en los clústeres.**
    - Algunos DBMS permiten configurar el tamaño que tendrán las paginas, el DBA encargado deberá hacerlo coincidir con el tamaño de clúster del SO.

- ***Fragmentación Interna***
  - Se produce cuando **lo que se va a almacenar en el cuerpo de la página es de menor tamaño que la longitud del renglón.**
  - **También puede** ocurrir que **el tamaño de la fila sea más grande que el tamaño del renglón de la página**, lo cual generara que **cada fila utiliza dos renglones o más para su almacenamiento.**



Pos	Id	Nombre	Apellido
0	123	CARLOS	RODRIGUEZ
1	204	PEDRO	ALBORNOZ
2	450	JOSE	MARTINEZ

Lógica de Almacenamiento

lunes, 5 de julio de 2021 04:00 p. m.

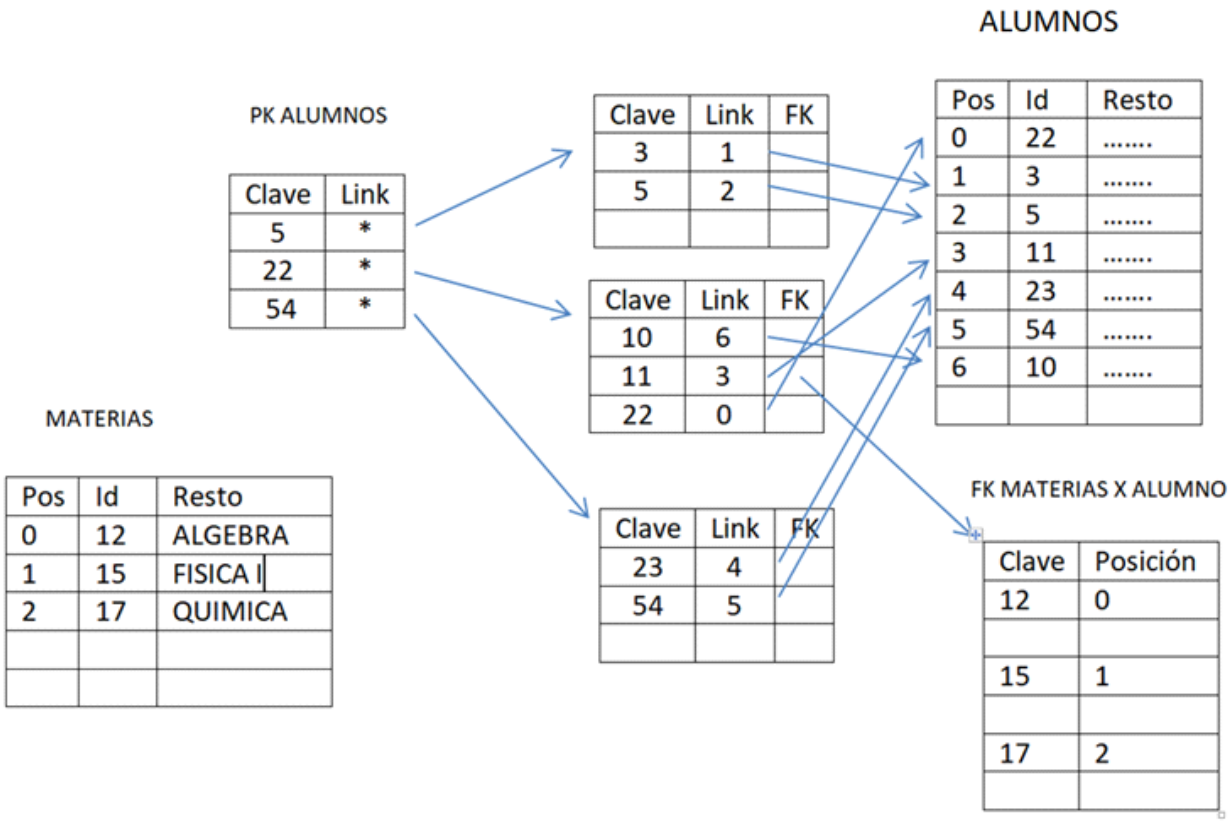
- **Clustering**
  - Es la técnica de agrupamiento que permite unificar objetos en función de algún criterio establecido.
  - Existen **2 formas** de aplicar esta técnica al momento de asignar datos:
    - ***Intra File (intra = dentro de un archivo)***
      - Los objetos **se agrupan en función** de la **pertenencia a un conjunto predeterminado**.
      - **Prioriza la pertenencia de la información de un objeto** para el almacenamiento, intenta almacenar en una página solo información de ese objeto.
    - ***Inter File (inter = relación entre archivos)***
      - Los objetos **se agrupan en función** a la **relación existentes entre** los **objetos independientemente que pertenezcan a diferentes conjuntos**.
      - **Prioriza las relaciones entre objetos,** intenta almacenar en la misma página solamente las relaciones entre los distintos objetos como pueden ser PK asociadas a las FK.

El DBMS utiliza ambas técnicas de almacenamiento simultaneamente para almacenar cosas distintas cuando le sea conveniente:

- ***Intra File***
  - Almacena de esta forma los datos secuenciales, o sea, en una página solo coloca filas que se corresponden a una tabla, sin mezclar tablas en la misma página. Entonces cuando quiera leer un SELECT * FROM de una tabla solo tendrá que leer una página.

- ***Inter File***
  - Almacena de esta forma los índices y PK asociadas a las FK que existan.

Almacenamiento de PK y FK (de manera Inter File)



PK Alumnos corresponde a un Árbol B de la tabla alumnos con punteros a las posiciones relativas de los datos (En ALUMNOS)

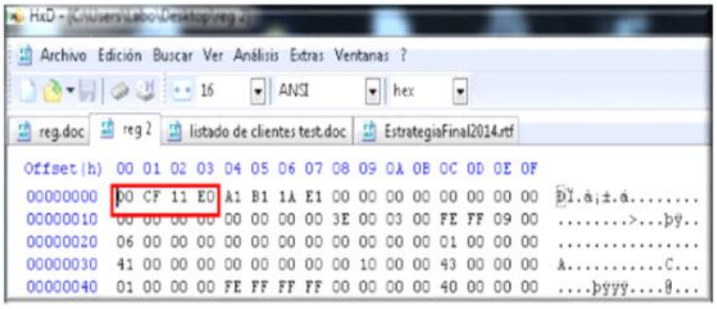
Al definir FK tiene que existir la PK por lo que agrega al árbol B los punteros (En columna FK) que apuntan a una tabla de hashing (MATERIAS X ALUMNO) donde están los campos relacionados al alumno, con las posiciones relativas a la posición de la tabla MATERIAS (columna Posición).

Cada hoja del árbol tiene una tabla de hashing, con todas las instancias que le corresponden a esa clave para no tener que ir a buscarlas, **se maneja solo con acceso directo, sin tener que buscar secuencialmente**.

Almacenamiento de Archivos (En una PC)

lunes, 5 de julio de 2021 04:02 p. m.

- ***Archivos***
  - Dado que el único formato de archivo existente y manejable por un sistema operativo es el compuesto por un conjunto de caracteres ASCII, **es necesario identificar** de alguna forma **el contenido de los mismos** para **poder tipificarlos y administrarlos** de forma diferencial.
- ***Header***
  - Conjunto de caracteres que se colocan al inicio del mismo y que **permiten definir el contenido** que continua en el mismo.
- ***Extensión***
  - La extensión de un archivo está relacionada con su **tipología y de hecho con la aplicación destinada a su apertura y administración**, **especifica el header** que se utilizará para **leer el contenido del archivo**.
  - De esta forma cada tipo de archivo comienza con alguna especificación diferente, lo cual permite identificarlo.



Tipo de Archivo	Cabecera	En ASCII
.ZIP	50 4B 03 04	PK
.RAR	52 61 72 21	Rar!
.DOC	D0 CF 11 E0	ÐÏ.à
.PDF	25 50 44 46	%PDF
.FLV	46 4C 56 01	FLV
.BMP	42 4D F8 A9/ 62 25 / 76 03	BM, BMp% , BMv
.GIF	47 49 46 38 39 61 / 37 61	GIF89a GIF87a
.JPEG	FF D8 FF E0 / FE	JFIF
.PNG	89 50 4E 47	PNG
.SFW	43 57 53 06 / 08	Cws
.MP3	49 44 33 2E /03	ID3
.EXE	4D 5A 50 00 /90 00	MZP / MZ
.DLL	4D 5A 90 00	MZ
Linux bin	7F 45 4C 46	ELF

Header de Tablas (DBMS)

lunes, 5 de julio de 2021 04:04 p. m.

El DBMS crea un header para identificar la estructura de las tablas y leerlas correctamente, siguiendo la idea de los headers que utilizan las PC para la lectura de los archivos:

```
struct table
{
        long filas;
        long columnas;
        int tamaño_fila;
};


struct columna
{
        char nombre[256];
        char tipo;
        int longitud;
        int decimales;
```

Los **tipos de y tamaños** de los **campos** de los **struct** deben ser **fijos** ya que el **header debe tener** una **estructura invariable** para su **correcta utilización**:

POS	CODIGO	NOMBRE	SALDO
0	05	CARLOS	12,35
1	03	MARIA	15,23
2	01	JOSE	18.89

```
struct table                                    VALORES
        {
                long filas;                     3
                long columnas;                  3
                int tamaño_fila;                64
        };

|
struct columna
        {
                char nombre[256];       CODIGO      NOMBRE      SALDO
                char tipo;              C           C           D
                int longitud;           2           50          12
                int decimales;          0           0           2
                boolean notnull;        1           0           0
        };
```

- El **struct table** indica que la tabla **tiene 3 filas (*long filas*)**, **3 columnas (*long columnas)*** y que el **tamaño ocupado por cada registro o fila** de la tabla es **64 bits (*int tamaño_fila),*** este valor se obtiene sumando el atributo *int longitud* de cada una de las columnas de la tabla.

- El **struct columna** describe las **caracteristicas del dato que se puede almacenar en cada columna**, como asi **tambien el nombre de la columna**.

    - El **nombre** de la **columna** se almacena en ***char nombre[256].***

    - El atributo ***char tipo*** indica el **tipo de dato** que se **puede almacenar en esa columna**
        - "C" indica tipo char en la columna CODIGO
        - "D" indica tipo decimal en la columna SALDO

- El atributo ***int longitud*** indica el **tamaño expresado** en bits que **ocupa** el **dato contenido** en dicha **columna**.

- El atributo ***int decimales*** indica la **cantidad de decimales** que tendrá el **contenido de la columna**.

- El atributo ***boolean notnull*** expresa **si el contenido de la columna puede ser NULL**.
  - Si es 1 NO permite almacenar valores NULL, en caso contrario si.