**Teoría de Grafos**

**Grafo** : Dado un conjunto de puntos P y un conjunto de relaciones E, la representación gráfica es lo que comúnmente denominamos grafo G(P, E).

Los grafos sirven para modelizar matemáticamente una estructura de datos. De este modelo, se pueden derivar resultados matemáticos teóricos.

**Grado**: cantidad de arcos salientes

**Maximales**: conjunto de nodos de los cuales no parte ningún arco con destino a otro nodo.

**Minimales**: conjunto de nodos a los cuales no le llega ningún arco proveniente de otro nodo. (si es un loop también es maximal)

**Right**: conjunto de nodos a los cuales llega un arco desde el nodo evaluado.

**Left**: conjunto de nodos desde los cuales sale un arco que llega al nodo evaluado.

**In degree:** cantidad de nodos que ingresa. (cantidad de arcos que ingresan)

**Out degree:** cantidad de nodos que salen. (cantidad de arcos que salen)

**Ideal derecho**: conjunto de nodos a los cuales se puede acceder (directa o indirectamente) desde el nodo evaluado (no puede ser conjunto vacío).

**Ideal izquierdo**: conjunto de nodos desde los cuales se puede acceder al nodo evaluado.

**Camino**: un camino entre a yb se produce cuando existe una vinculación directa o indirecta entre ambos, esto es cuando puedo vincular mediante uno o más arcos dichos nodos entre sí. Puede existir tanto en grafos dirigidos como en grafos no dirigidos.

**Walk**: es un camino sin dirección.

**Paso:** un paso entre dos nodos a y b, se produce cuando existe un camino entre ambos, pero con un sentido preestablecido (existe una secuencia de nodos para llegar a otro nodo).

**Paso simple:** todos los nodos son distintos (no hay loop)

**Longitud de paso:** cantidad de arcos para llegar al nodo.

**Ciclo:** es un camino donde no se recorre dos veces el mismo arco y donde regresa al punto inicial (ciclo hamiltoniano).

**Grafo simple:** un grafo es simple si a lo sumo un arco une dos nodos cualesquiera.

**Grafo complejo:** es un grafo que no es simple:

**Grafo conexo:** un grafo es conexo si cada par de nodos está conectado por un camino.

**Grafo completo:** un grafo simple es completo si existen arcos uniendo todos los pares posibles de nodos.

**Grafo bipartito:** es un grafo en el cual el conjunto de puntos se divide en dos subconjuntos disjuntos, de tal manera que no exista un arco entre dos nodos que pertenezcan a un mismo conjunto.

**Grafos dirigidos:** son aquellos en los cuales los arcos que vinculan a los nodos tienen una dirección definida.

**Grafos no dirigidos:** son aquellos donde los arcos no tienen una dirección definida.

**Grafos restrictos:** son aquellos en los cuales la relación modelizada no debe cumplir las propiedades reflexivilidad, simetría y transitividad.

**Grafos irrestrictos:** son aquellos que pueden modelizar cualquier relación independientemente de las propiedades que cumpla o no.

**Grafo fuertemente conectado:** existe paso entre dos nodos cualquiera. Para cualquier

par de nodos x,z ∈ P, existe δ(x, z).

**Grafo acíclico:** un grafo es acíclico cuando cumple con la relación de orden parcial.

Relación de orden parcial:

- Antisimétrica

- Reflexiba

- Transitividad

Relación de orden total:

**Clausura transitiva:** (pag 20)

Así como por definición, en el grafo básico se omiten los arcos, la clausura transitiva

de un grafo surge como consecuencia de agregar al grafo original los arcos de las relaciones resultantes al obtener la relación de paso, mostrando en el grafo todas las relaciones posibles.

**Algoritmo de Warshall:** (pag 22)

1) Para conocer la cantidad de pasos distintos de igual longitud hay entre dos nodos.

2) Para saber la longitud de paso mínima entre dos nodos.

El algoritmo de Warshall nos permite obtener más información que la matriz de la clausura transitiva, utilizando una técnica similar. En lugar de usar una multiplicación booleana se utiliza la multiplicación aritmética de matrices.

*Matriz de Warshall:* nos dice si hay relación entre dos nodos y cuantos pasos mínimamente llego a él (longitud de paso mínima).

**Algoritmo de Siklóssy:**

Este algoritmo presenta la posibilidad de minimizar el espacio ocupado, utilizando un único campo link, sin aumentar el tiempo de acceso, mediante el concepto algebraico del or exclusivo.

Para evitar el doble linkeo, Sikloosy propone almacenar en un mismo campo de dirección un valor que permita recorrer la estructura en ambos sentidos basándose en la suma del or exclusivo.

**Representación de Pfaltz:**

Pfaltz trabaja con una representación dinámica de nodos y arcos enlazados mediante punteros, ocupando espacio en memoria a medida que realmente se necesita (cuando se crea un nuevo nodo o un nuevo arco), y liberándolo cuando se efectúa una baja.

Para un problema medianamente complejo donde se pueden llegar a tener cientos de nodos, la

representación de Pfaltz es más eficiente que el uso de matrices. Una matriz sería extremadamente grande teniendo en cuenta que siempre se reserva espacio para las relaciones, independientemente de la cantidad de arcos realmente existentes.

---------------------------------------------------------------------------------------------------------------------------

**Formas de aplicar el grafo computacionalmente**

* Estáticas
  + Matriz de adyacencia
  + Matriz de incidencia
* Dinámicas
  + Listas de adyacencia
  + Estructura de Graal
  + Estructura de Pfaltz
* Mixtos
  + Vector arcos entrantes
  + Vector arcos salientes

---------------------------------------------------------------------------------------------------------------------------

**Representación Dinámica:**

Una representación es dinámica cuando el espacio consumido para representar computacionalmente al grafo, concuerda exactamente con la cantidad de nodos y arcos a representar, esto es que no se consideran todas las posibilidades de relación posibles, sino que solo se representa lo que ocurre en ese momento.

Dinámicamente un grafo irrestricto puede representarse a través de estructuras como:

* Listas de adyacencia
* Estructura de Graal
* Estructura de Pfaltz

**Representación Estática:**

Una representación es estática cuando el espacio consumido para representar computacionalmente el grafo es invariable y fijo respecto a la cantidad de nodos y arcos a representar, esto es que son consideradas todas las ocurrencias de relaciones que pueden producirse entre todos los nodos, reservando el espacio para dicha ocurrencia potencial.

Estáticamente un grafo irrestricto puede representarse a través de :

* Matriz de adyacencia
* Matriz de incidencia

---------------------------------------------------------------------------------------------------------------------------

**Complejidad espacial y computacional.**

En los sistemas de datos se tiende a medir los algoritmos en términos de tiempo de procesador utilizado (complejidad computacional) y en espacio (en disco o memoria) utilizado (complejidad espacial). De esta manera un algoritmo será mejor que otro si utiliza menos recursos ya sea en tiempo de procesador utilizado o en memoria.

Complejidad espacial -> espacio físico

vs

Complejidad computacional -> tiempo de procesamiento

---------------------------------------------------------------------------------------------------------------------------

**Árboles**

**Un árbol es un grafo que cumple lo siguiente**:

* Es acíclico
* Sumatoria de nodos = Sumatoria de arcos + 1
* Todo arco es desconectante
* Existe Walk único para todo par de nodos

**Subárbol**: conjunto de nodos y arcos que en sí mismo son un árbol.

**Subárbol Principal Derecho**: conjunto de nodos a los cuales se puede llegar desde el nodo evaluado y sus arcos.

**Subárbol Principal Izquierdo:** conjunto de nodos desde los cuales se puede llegar al nodo evaluado y sus arcos.

**Árbol Principal Derecho**: árbol donde existe un único nodo cuyo SAPD es todo el árbol.

**Árbol Principal Izquierdo**: árbol donde existe un único nodo cuyo SAPI es todo el árbol.

**Grado del árbol**: está dado por el grado de salida del nodo con mayor grado de salida.

**Profundidad de un nodo**: distancia entre un nodo y la raíz.

**Nivel**: profundidad más alta.

**Árbol completo**: árbol donde todos los nodos que no son hojas tienen el mismo grado.

**Árbol lleno**: árbol completo que tiene a todos sus hojas en el mismo nivel.

Árbol lleno = Árbol balanceado ???

**Barridos**

* **Preorden:** raiz-izq-der
* **Simétrico**: izq-raiz-der
* **Posorden**: izq-der-raiz
* **Por niveles**

**Árbol de expresión**

* Binario
* Completo
* Las hojas son lo operandos
* Las no hojas son los operadores

**Transformada de KNUTH**

Knuth transforma un árbol r-ario en un árbol binario de forma tal que, dado un nodo, coloca como hijo izquierdo el primer elemento de su right y como hijo derecho el siguiente nodo

del mismo nivel del mismo padre.

**Árbol - B**

Por como es su estructura, siempre va quedando balanceado. Consta de n-1 claves y n arcos.

En cada hoja, pongo varias claves(y varios punteros). Estas hojas apuntan a los datos.

Están pensados para disminuir la cantidad de accesos a disco y la posibilidad de mantener en memoria lo que se está utilizando y en disco el resto. Se utiliza para grandes volúmenes de datos y es mejor para archivos secuenciales. Árbol B es más lento que Hashing para la creación de índices ya que para ello debe crear toda la estructura en memoria.

Lo costoso de este árbol es al hacer el split, que puede generar varios splits sucesivos. Existe también el Load Factor, es una reserva de espacio para poder cargar un valor y no tener que hacer split. Se puede ir jugando con ese % de Load Factor para mejorar la performance.

---------------------------------------------------------------------------------------------------------------------------

**Ordenamiento y búsqueda**

**Heapsort**

Es un algoritmo de ordenamiento que utiliza como estructura auxiliar un árbol binario donde establece un orden parcial entre sus nodos, en donde el padre será mayor que sus hijos. El orden de complejidad de carga y ordenamiento, al ser un árbol binario, es el mismo n.log2(n+1)

Es más eficiente si los datos vienen ordenados.

**Quicksort**

Es un algoritmo de ordenamiento con partición. Está basado en la técnica de divide y vencerás, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n.log(n). El algoritmo original es recursivo, pero se utilizan versiones iterativas para mejorar su rendimiento (los algoritmos recursivos son en general más lentos que los iterativos, y consumen más recursos.)

Si los datos vienen ordenados es ineficiente, el orden es n2, y si los datos vienen desordenados es más eficiente, orden de n.log2(n).

**Árbol Binario** (no es lo mismo que Arbol B)

Es un algoritmo de ordenamiento que utiliza un árbol binario. En este método todos los descendientes izquierdos son menores y los derechos son mayores. El recorrido simétrico del árbol genera la lista ordenada.

La ventaja de utilizar este método es que un árbol permite la búsqueda, inserción y eliminación en forma eficiente; un vector requeriría que la mitad de los elementos sean movidos; por el otro la inserción o eliminación en árbol binario de búsqueda, requiere que se ajusten unos pocos puteros.

Funciona mejor cuando los datos están desordenados

Es difícil mantenerlo balanceado, los motores usan Árbol B o B+

* *Árbol balanceado:* es un árbol binario de búsqueda donde se cumple que para todo nodo del árbol, la altura de los subárboles izquierdo y derecho no debe ser más de 1. Llamaremos punto crítico al nodo en el cual el árbol se desbalancea.
* Árbol AVL:

Para el l Balanceado las salidas de cada nodo hijo de una raíz deben dar una diferencia <= 1, en AVL lo que se establece es una función de altura, donde la diferencia de altura es <= 1.

Uno mide la cantidad de salidas (outdegree) y el otro la altura.

El orden lo va a dar el recorrido simétrico al árbol.

---------------------------------------------------------------------------------------------------------------------------

**Algoritmo de Huffman**

Es un algoritmo que puede ser usado para *compresión o encriptación de datos*. Por ej: se usa cuando la cantidad de espacio en disco es insuficiente o los tiempos de transmisión son prolongados con costos elevados.

Huffman se basa en asignar códigos de distinta longitud de bits a cada uno de los caracteres de un archivo. Si se asignan códigos más cortos a los caracteres que aparecen más a menudo se consigue una compresión del archivo.

Esta compresión es mayor cuando la variedad de caracteres diferentes que aparecen es menor. Ej: si el texto se compone únicamente de números o mayúsculas, se conseguirá una compresión mayor.

Para recuperar el archivo original es necesario conocer el código asignado a cada caracter, así como su longitud en bits, si esta información se omite, y el receptor del archivo la conoce, podrá recuperar la información original. De este modo es posible utilizar el algoritmo para encriptar archivos.

Huffman hace uso de una tabla de frecuencias donde se guardan los caracteres empleados en el texto y las cantidad de veces que son utilizados y de un árbol binario como estructura de datos.

El código generado por el algoritmo está basado en el código Morse y se define una longitud variable de codificación basada en estadísticos.

---------------------------------------------------------------------------------------------------------------------------

**Conceptos de Base de Datos**

**Base de datos**

Una BD es un conjunto de datos persistentes e interrelacionados que es utilizado por los sistemas de aplicación de una empresa, los mismos se encuentran almacenados en un conjunto independiente y sin redundancia.

**Componentes de un Sistema de BD** (pag 3)

* Datos
* Tecnología
* Programas/Procesos
* Usuarios

**DBMS - Sistema de Administración de BD o Motor de BD**

Es un programa que permite administrar los contenidos de una BD almacenada en disco e interpreta y ejecuta todos los comandos SQL.

Componentes principales de un DBMS

* Procesos:
  + Controlan el motor de BD
  + Proveen funcionalidades específicas asociadas a seguridad, integridad, operación, organización interna entre otros.
* Memoria compartida: *es usada para*
  + Cachear datos del disco para tener un acceso más rápido.
  + Mantener y controlar los recursos.
  + Proveer mecanismos de comunicación para los procesos clientes o propios del motor.
* Unidades de disco
  + Es la colección de una o más unidades de espacio de disco asignadas al DBMS.

Enfoques

* **Enfoque de aplicaciones tradicionales**

Cada programa manipula a nivel físico los archivos, peticiones al S.O. Las operaciones de lectura/escritura sobre los mismos. El desarrollador debe tener en cuenta durante el diseño y construcción factores relativos tales como concurrencia entre aplicaciones.

* **Enfoque de Base de Datos**

Los archivos de datos (tablas) residen en la BD y los programas o consultas de usuarios son enviadas mediante algún mecanismo de comunicación al DBMS, que es quien realiza realmente la consulta/actualización y devuelve el resultado al programa. Las peticiones no se hacen al S.O. sinó al DBMS mediante instrucciones SQL.

* Ventajas
  + Seguridad
  + Estándares de documentación y normalización de nomenclaturas
  + Redundancia mínima
  + Consistencia de datos (transacciones)
  + Concurrencia en acceso a datos
  + Integridad de los datos (restricciones, objetos de BD)
  + Criterios y normativas para organización de datos
  + Independencia
* Independencia de datos
  + *Independencia lógica:* cada aplicación requiere una vista diferente de los mismos datos y no requiere saber todos los atributos de una tabla o el orden real de dichos atributos, ni la distribución de atributos a través de las tablas. Se podrían hacer cambios en las
  + *Independencia física:* es posible modificar la estructura de almacenamiento, la distribución física o la técnica de acceso sin afectar las aplicaciones. (crear un índice para que se resuelva más rápidamente, crear un objeto determinado asociado a una tabla, etc.)

**Arquitectura de un Sistema de Base de Datos - ANSI SPARC**

* *Nivel externo (VISTAS):* esta es la percepción que tienen los usuarios de la BD. Los usuarios pueden ser programadores, usuarios finales, o el DBA.
* *Nivel conceptual (LÓGICO):* representa, de una forma entendible, toda la información contenida en una BD. Este esquema se escribe en DDL, contiene definiciones de la base, tipos de datos, restricciones, reglas de integridad, etc.
* *Nivel interno (FÍSICO):* en este nivel se define cómo se almacenan los datos en disco. Se especifican las estructuras internas de disco y memoria, métodos de acceso, etc.

**Funciones del Motor de Base de Datos y del DBMS en su conjunto**

1. *Diccionario de datos:* es un conjunto de datos del sistema, que define cada uno de los objetos dentro del mismo. Es lo que se llama “metadatos”.
2. *Control de la seguridad:* la seguridad implica controlar que los usuarios estén autorizados para hacer lo que intentan. Para esto el DBMS tiene un “catálogo” compuesto de tablas con entidades y relaciones.
3. *Mecanismos para garantizar la integridad de datos:* para esto el DBMS cuenta con distintas restricciones y objetos que pueden ser creados y procesos que se encargan de controlar que se cumplan dichas restricciones.
4. *Mecanismos para garantizar la consistencia de datos:* conceptos relacionados con la consistencia de datos:

* Transacciones: es un conjunto de sentencias SQL que se ejecutan atómicamente en una unidad lógica de trabajo.
* Logs transaccionales: es un registro donde el motor almacena la información de cada operación llevada a cabo con los datos.
* Recovery: método de recuperación ante caídas.

1. *Mecanismos de resguardo y restauración (BACKUPS Y RESTORE):*

* Backup: es una copia total o parcial de la información de una BD.
* Restore: es la acción de tomar un backup y restaurar la estructura sobre una BD.

1. *Mecanismos de recuperación (RECOVERY):* es un mecanismo provisto por los motores de BD automático que se ejecuta en cada inicio del motor de forma automática. Su objetivo es retornar al Motor de BD al punto consistente más reciente.
2. *Facilidades de auditoría:* a los DBMS se le puede activar la opción de guardar en un log un registro del uso de los recursos de la base para auditar posteriormente.
3. *Logs del sistema:* mediante este tipo de logs, el DBA puede llegar a determinar cuál fué, por ejemplo, el problema que produjo la caída del sistema.
4. *Acomodarse a cambios, crecimientos, modificaciones del esquema:* el motor permite realizar cambios a las tablas constantemente.
5. *Creación de triggers & stored procedures*

*----------------------------------------------------------------------------------------------*

**Recuperación y acceso a datos**

**Algoritmos de recuperación**

Estos algoritmos pueden devolver un registro completo o un puntero a ese registro. Deben contemplar la posibilidad de insertar una clave única, con lo cual deberá verificar que la clave no exista previamente o de recuperar una clave validando que la misma exista.

* *Búsqueda secuencial:* consiste en efectuar una lectura secuencial del archivo hasta detectar que el valor que tiene ciertos campos coinciden con un argumento de búsqueda. En caso de no hallarse se detectará recién después de haber leído todo el archivo. Orden de complejidad n/2.
* *Búsqueda secuencial indexada:* esta técnica mejora el tiempo de recuperación, pero al hacer uso de una tabla externa ordenada es necesario disponer de más espacio físico. Cada elemento de la tabla contiene un valor de clave y la dirección donde se encuentran los datos de esa clave.
* Hashing:
* Índices
* Hashing
  + Módulo
  + Método de dobles
  + Método del cuadrado
  + Colisiones
  + Hash into bucket: dentro de la tabla de Hash se le asocia una lista o vector, cuando hay mas de un valor que tiene el mismo Hash (colisión) pag 8

---------------------------------------------------------------------------------------------------------------------------

**Data Warehousing**

Los sistemas de Data Warehousing son el centro de la arquitectura de los Sistemas de Información de los 90's. Han surgido como respuesta a la problemática de extraer información sintética a partir de datos atómicos almacenados en bases de datos de producción. Uno de los objetivos principales de este tipo de sistemas es servir como base de información para la

toma de decisiones. Los beneficios obtenidos por la utilización de este tipo de sistemas se basan en el acceso interactivo e inmediato a información estratégica de un área de negocios. Este acercamiento de la información al usuario final permite una toma de decisiones rápida y basada en datos objetivos obtenidos a partir de las bases de datos (eventualmente

heterogéneas) de la empresa. Estos beneficios aumentan cuanto más importantes son las decisiones a tomar y cuanto más crítico es el factor tiempo.

Un Sistemas de Data Warehousing incluye funcionalidades tales como:

1. Integración de bases de datos heterogéneas (relacionales, documentales, geográficas, archivos, etc.).

2. Ejecución de consultas complejas no predefinidas visualizando el resultado en forma de gráfica y en diferentes niveles de agrupamiento y totalización de datos.

3. Agrupamiento y desagrupamiento de datos en forma interactiva.

4. Análisis de problema en términos de dimensiones. Por ejemplo, permite analizar datos históricos a través de una dimensión tiempo.

5. Control de calidad de datos para asegurar, no solo la consistencia de la base, sino también la relevancia de los datos en base a los cuales se toman las decisiones.

**Data Mining**

Data Mining, la extracción de información oculta y predecible de grandes bases de datos, es una poderosa tecnología nueva con gran potencial para ayudar a las compañías a concentrarse en la información más importante de sus Bases de Información (Data Warehouse). Las herramientas de Data Mining predicen futuras tendencias y comportamientos, permitiendo

en los negocios tomar decisiones proactivas y conducidas por un conocimiento acabado de la información (knowledge-driven). Los análisis prospectivos automatizados ofrecidos por un producto así van más allá de los eventos pasados provistos por herramientas retrospectivas típicas de sistemas de soporte de decisión. Las herramientas de Data Mining pueden

responder a preguntas de negocios que tradicionalmente consumen demasiado tiempo para poder ser resueltas y a los cuales los usuarios de esta información casi no están dispuestos a aceptar. Estas herramientas exploran las bases de datos en busca de patrones ocultos, encontrando información predecible que un experto no puede llegar a encontrar porque se encuentra fuera de sus expectativas.

**OLAP vs OLPT**

Las aplicaciones OLTP se caracterizan por la creación de muchos usuarios, actualizaciones o recuperación de registros individuales. Por consiguiente, las bases de datos OLTP se perfeccionan para actualización de transacciones. Las aplicaciones OLAP son usadas por analistas y gerentes que frecuentemente quieren una vista de datos de nivel superior, como las ventas totales por línea de producto, por región, etc. Las bases de datos OLAP normalmente se actualizan en lote, a menudo de múltiples fuentes, y proporcionan un back-end analítico poderoso a las aplicaciones de múltiples usuarios. Por tanto, las bases de datos OLAP se perfeccionan para el análisis.

Mientras las bases de datos relacionales son buenas al recuperar un número pequeño de archivos rápidamente, ellas no son buenas al recuperar un número grande de archivos y resumirlos on-the-fly. Un tiempo de respuesta lento y el uso excesivo de recursos del sistema son las características comunes de las aplicaciones de soporte de decisión construidas exclusivamente sobre la tecnología de bases de datos relacionales. Debido a la facilidad con la cuál se puede emitir un “ejecutar una consulta SQL externa”, muchos distribuidores IS (Information Systems) no brindan acceso directo a los usuarios a sus bases de datos relacionales.

Muchos de los problemas que las personas intentan resolver con la tecnología relacional son realmente multidimensionales en naturaleza. Por ejemplo, una consulta SQL para crear resúmenes de ventas del producto por la región, las ventas de la región por producto, y así sucesivamente, podrían involucrar la revisión de la mayoría, si no todos, de los registros en una base de datos de mercadeo y podría tomar horas de proceso. Un servidor OLAP podría ocuparse de estas preguntas en unos segundos.

|  |  |
| --- | --- |
| OLTP (Relational) | OLAP(Multidimensional) |
| Atomized  Present  Record-at-a-time  Process oriented | Summarized  Historical  Many records at a time  Subject oriented |

Las aplicaciones OLTP tienden a tratar con datos atomizados “registro a un tiempo”, considerando que las aplicaciones de OLAP normalmente se tratan de los datos resumidos. Mientras las aplicaciones OLTP generalmente no requieren de datos históricos, casi cada aplicación de OLAP se preocupa por ver las tendencias y por consiguiente requiere de datos históricos. Como consecuencia, las bases de datos OLAP necesitan la capacidad de ocuparse de datos series de. Mientras las aplicaciones OLTP y bases de datos tienden a ser organizados alrededor de procesos específicos (como órdenes de entrada), las aplicaciones OLAP tienden a ser “orientadas al tema”, respondiendo a preguntas como “¿Qué productos están vendiendo bien?” o “¿Dónde están mis oficinas de ventas más débiles?”.