Definición, metodologías y mejores prácticas para la optimización

Optimización

- La optimización de una base de datos tiene como objetivo reducir el tiempo de respuesta del sistema, maximizando la velocidad y eficiencia con la que recuperan los datos, mejorando la experiencia del usuario final.
- Para lograr una buena optimización es necesario contar con un diseño físico de base de datos eficiente.
- Para ser más eficiente en el diseño físico de bases de datos es necesario llevar a cabo el proceso del diseño físico de bases de datos incluyendo entradas, salidas y objetivos, junto con dos conceptos críticos del entorno: estructuras de archivos y optimización de consultas.

Diseño físico de las bases de datos: Generalidades

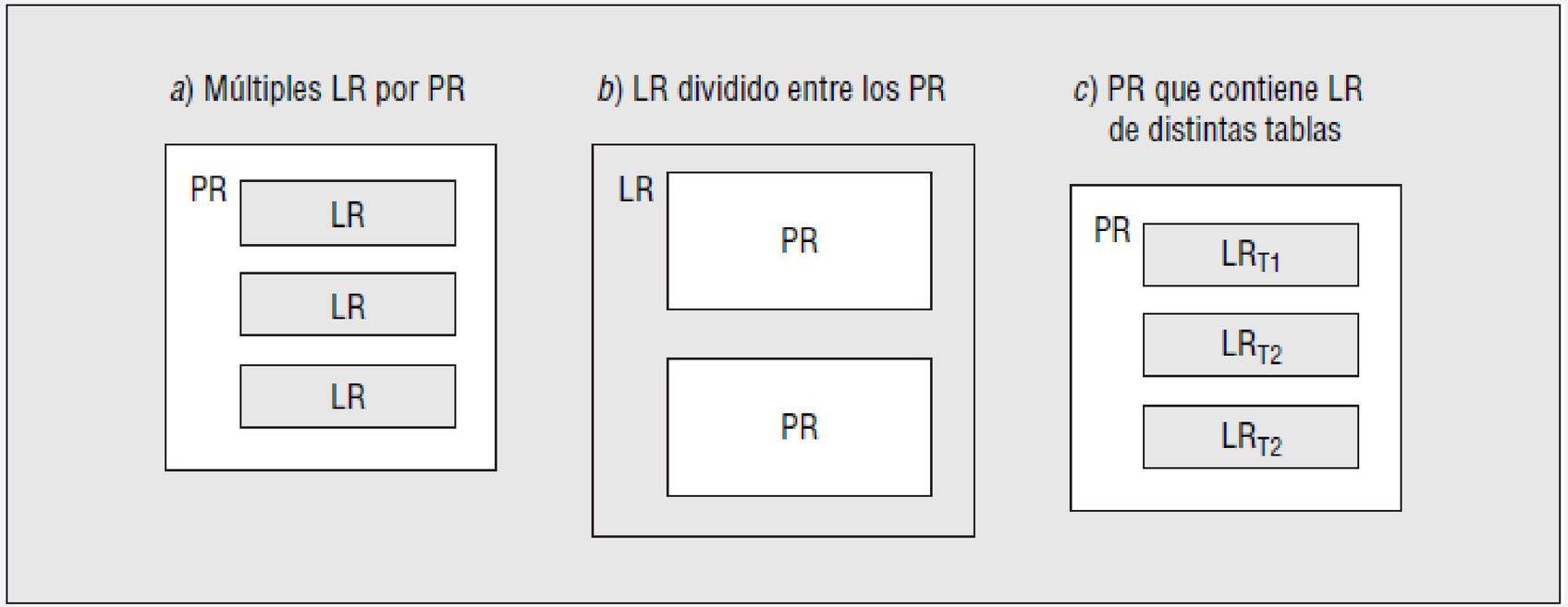
- Las decisiones en la fase del diseño físico de bases de datos involucran el nivel de almacenamiento de una base de datos (esquema interno).
- El objetivo del diseño físico de bases de datos es minimizar los tiempos de respuesta para acceder y modificar una base de datos

Diseño físico de las bases de datos: Generalidades (cont.)

- El nivel de almacenamiento:
  - Es el más cercano al *hardware* y al sistema operativo.
  - La base de datos está formada de registros físicos (también conocidos como bloques o páginas) organizados en archivos.
  - Un registro físico es un conjunto de bytes que se transfieren entre el almacenamiento volátil de la memoria principal y el almacenamiento fijo de un disco.
  - A la memoria principal se le considera como almacenamiento volátil porque los contenidos de la memoria principal se pueden perder si ocurre alguna falla.
  - Un archivo es un conjunto de registros físicos organizados para conseguir un acceso eficiente.
  - Un registro físico puede contener varios registros lógicos.
  - El tamaño de un registro físico es una potencia del número dos, por ejemplo 1 024 (2 potencia 10).
  - Un registro lógico muy grande se puede dividir entre varios registros físicos.
  - Los registros lógicos de más de una tabla se pueden almacenar en el mismo registro físico.
  - El DBMS y el sistema operativo trabajan de manera conjunta para satisfacer las solicitudes de registros lógicos hechas por las aplicaciones.

Diseño físico de las bases de datos: Generalidades (cont.)

- Relaciones entre los registros lógicos (RL) (filas de una tabla) y registros físicos (PR) almacenados en un archivo.



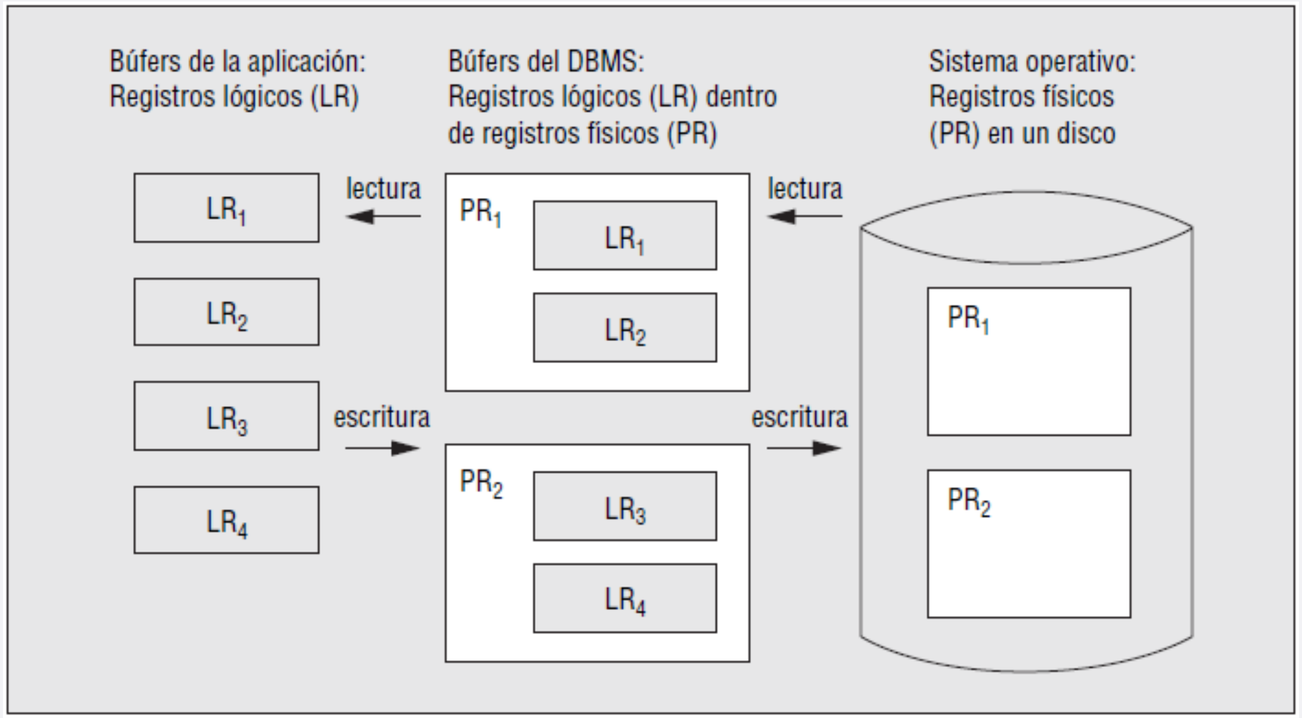
Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.1, pp. 250), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- El DBMS y la aplicación tienen áreas de memoria separadas (*buffers*).
- Cuando una aplicación hace una solicitud para un registro lógico, el DBMS ubica al registro físico que lo contiene.
- Para las operaciones de lectura, el sistema operativo transfiere el registro físico del disco al área de memoria del DBMS.
- El DBMS transfiere el registro lógico al *buffer* de la aplicación.
- Para las operaciones de escritura, el proceso de transferencia se invierte.
- Un requerimiento de registro lógico no siempre resulta en una transferencia de registro físico debido al proceso denominado *buffering*, así el DBMS se anticipa a las solicitudes de las aplicaciones para que los registros físicos correspondientes residan ya en sus *buffers*.
- Una dificultad significativa acerca de la predicción del desempeño de la base de datos es conocer cuándo una solicitud de registro lógico conduce a una transferencia de registro físico.
- Si varias aplicaciones están accediendo a los mismos registros lógicos, los registros físicos correspondientes pueden residir en los *buffers* del DBMS, resultando en una posible dificultad durante el diseño físico de la base de datos.

Diseño físico de las bases de datos: Generalidades (cont.)

- Proceso de transferencia de registros físicos (PR) y registros lógicos (LR) entre un disco, *buffers* del DBMS y *buffers* de la aplicación.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.2, pp. 251), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

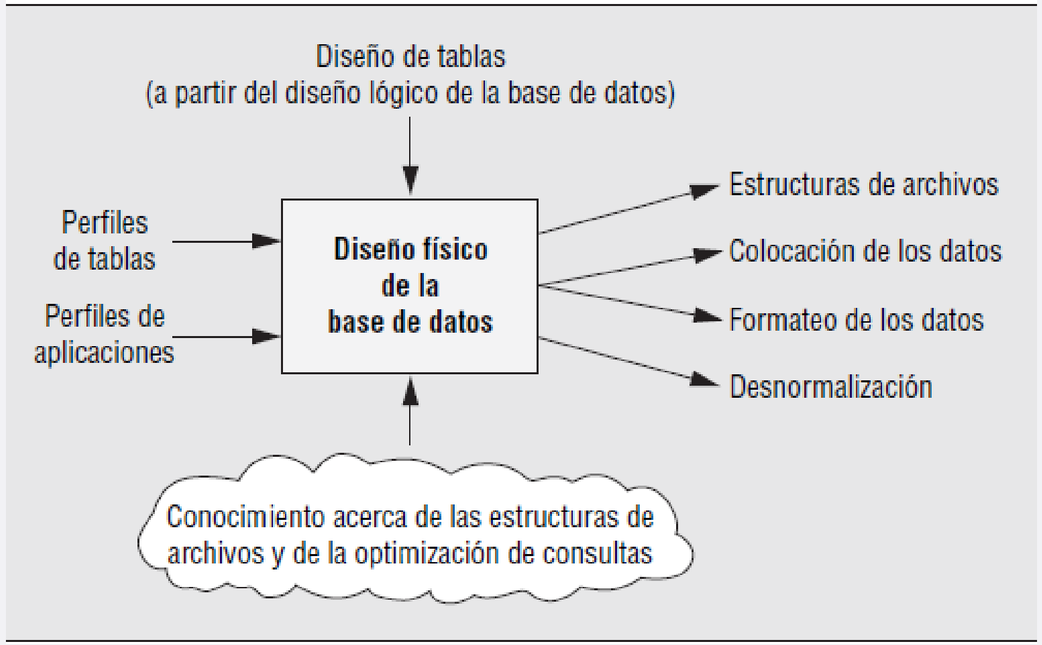
- El tiempo de respuesta es difícil de estimar de modo directo y como medida sustituta se aplica la minimización de los recursos de cómputo.
- Los recursos que consume el procesamiento de la base de datos son la transferencia de registros físicos, las operaciones de la unidad central de procesamiento (CPU), la memoria principal y el espacio en disco.
- La memoria principal y espacio en disco son considerados como restricciones, en lugar de recursos a minimizar.
- La minimización de la memoria principal y del espacio en disco puede ocasionar tiempos largos de respuesta. El número de accesos a los registros físicos limita el desempeño de la mayoría de las aplicaciones de bases de datos.
- El acceso a un registro físico (medido en milisegundos) puede ser muchas veces más lento que el acceso a la memoria principal (medido en nanosegundos).
- La reducción del número de accesos a registros físicos mejorará el tiempo de respuesta.
- El uso de CPU es otro factor que influyen en los tiempos de respuestas aplicaciones de bases de datos.

Diseño físico de las bases de datos: Generalidades (cont.)

- Se utiliza un peso cercano a 0 para reflejar que muchas operaciones del CPU se pueden realizar en el tiempo en el que se realiza una transferencia de registro físico.
- En el diseño físico de bases de datos es muy importante balancear las necesidades de recuperación y actualización de las aplicaciones, siendo usualmente fijas las cantidades de memoria principal y de espacio en disco.
- La memoria principal y el espacio en disco son restricciones del proceso de diseño físico de bases de datos, por lo que deben considerarse los efectos ocasionados cuando se modifican las cantidades proporcionadas de memoria principal y espacio en disco, un incremento de las cantidades puede mejorar el desempeño.
- La cantidad de mejora del desempeño puede depender de muchos factores tales como el DBMS, el diseño de las tablas y las aplicaciones que usa la base de datos.
- El *software* de optimización compleja generalmente es parte del compilador de SQL.

Diseño físico de las bases de datos: Generalidades (cont.)

- El diseño físico de bases de datos está formado por varias entradas y diferentes salidas.
- El punto de partida es el diseño de tablas a partir de la fase del diseño lógico.
- Los perfiles de las tablas y aplicaciones se usan específicamente para el diseño físico de bases de datos.
- Las salidas más importantes son las decisiones de las estructuras de archivos y la colocación de los datos, las decisiones acerca de otras salidas se hacen de forma separada, a pesar de que estén relacionadas



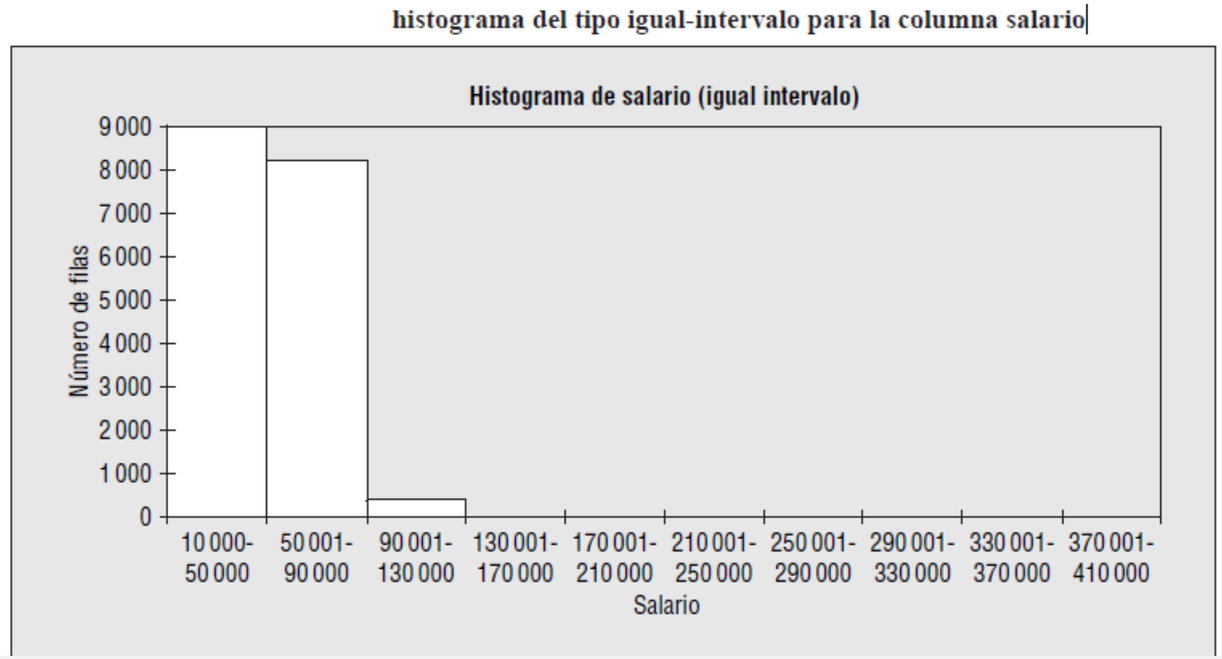
Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.3, pp. 252), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- El diseño físico de bases de datos es una secuencia de procesos de toma de decisiones.
- El conocimiento de las estructuras de archivos y la optimización de consultas están dentro del entorno del diseño físico de bases de datos.
- El diseño físico de las bases de datos se puede dificultar debido al número de decisiones, relaciones entre las decisiones, entradas detalladas, complejidad del entorno e incertidumbre para predecir los accesos a los registros físicos.
- El diseño físico de las bases de datos requiere de entradas detallas tanto para los perfiles de las tablas como los perfiles de las aplicaciones.
- El perfil de una tabla resume una tabla como un todo, las columnas dentro de la tabla y la relación entre las tablas. La mayoría de los DBMS proporcionan programas estadísticos para construir perfiles de tabla de forma automática, se suele establecer la periodicidad de ejecución de esos programas. Para bases de datos muy grandes, los perfiles de las tablas se pueden estimar con ejemplos de la base de datos, para reducir el tiempo insumido por esta tarea.
- Para los resúmenes de columnas y relaciones, la distribución expresa el número de filas y filas relacionadas para los valores de las columnas. Se suele asumir que los valores de las columnas están distribuidos de modo uniforme, es decir que cada valor tiene un número igual de filas y solo se necesitan los valores mínimo y máximo.

Diseño físico de las bases de datos: Generalidades (cont.)

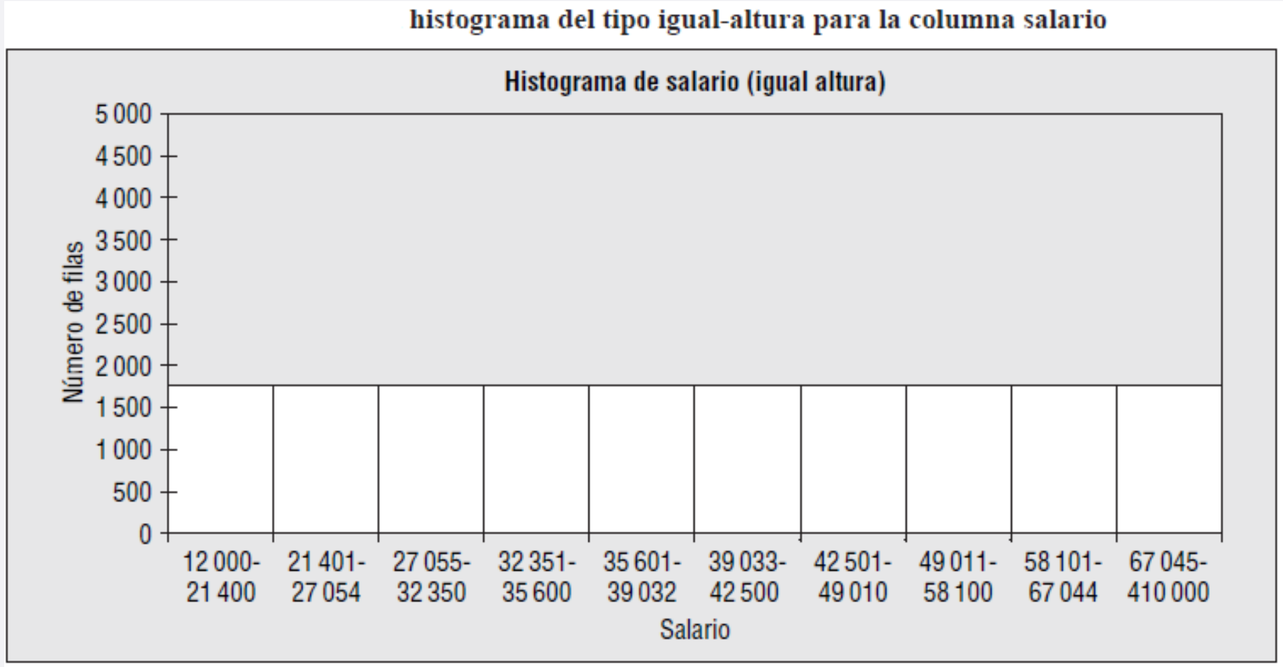
- Histogramas:
  - Se utilizan para especificar con más detalle la distribución.
  - Es un gráfico de dos dimensiones en la que el eje X representa los rangos de las columnas y el eje Y representa el número de filas.
- Tipos de histogramas:
  1. Igual intervalo para los valores de una columna



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.4, pp. 254), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- Histogramas:
  - Los histogramas tradicionales del mismo intervalo no funcionan bien con datos asimétricos, ya que es necesario un gran número de rangos para controlar los errores estimados.
  - Debido a que los datos asimétricos pueden conducir a estimaciones pobres con el uso de los histogramas tradicionales (intervalo igual), la mayoría de los DBMS utilizan histogramas del tipo igual-altura.
- 2. Igual altura para los valores de una columna



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.5, pp. 255), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- Histogramas:
  - En los histogramas de igual-altura para los valores de una columna, la anchura de los rangos varía, pero la altura es aproximadamente la misma.
  - La mayoría de los DBMS usan histogramas del tipo igual-altura, dado que los errores de estimación máximos y esperados se pueden controlar aumentando el número de rangos.
  - Generalmente el número de registros físicos se usa para calcular los accesos a registros físicos para extraer las filas de una tabla.
  - La distribución de los valores de las columnas es necesaria para estimar la fracción de filas que satisfacen una condición en una consulta.
  - Es de gran utilidad guardar datos estadísticos de columnas que se encuentren relacionadas para evitar caer en errores, cuando se estime la fracción de filas que satisfacen las condiciones que conectan las columnas a través de los operadores lógicos (AND, OR).

Diseño físico de las bases de datos: Generalidades (cont.)

- Los perfiles de las aplicaciones resumen las consultas, formularios y reportes que acceden a una base de datos.
- Para los formularios se debe especificar la frecuencia del uso del formulario para cada operación (inserción, actualización, eliminación y recuperación).
- Para las consultas y reportes, la distribución de los valores de los parámetros codifica el número de veces que se ejecuta la consulta/reporte con varios valores para los parámetros.
- La frecuencia de los datos se especifica como un promedio de la unidad de tiempo, por ejemplo un día. Se pueden resumir las frecuencias con más detalle, por ejemplo, considerando las frecuencias pico y la varianza de las frecuencias.
- Clasificar las aplicaciones según su importancia, por ejemplo considerando los tiempos de respuesta límite, puede ser de utilidad para que los diseños físicos de las bases de datos tengan cierta parcialidad hacia las aplicaciones críticas.

Diseño físico de las bases de datos: Generalidades (cont.)

- Otro aspecto fundamental dentro del diseño físico de las bases de datos es la selección de la estructura de los archivos.
- Las estructuras de archivos disponibles son:
  - Archivos secuenciales: tienen una organización simple en la que los registros se almacenan en el orden de inserción o mediante el valor de una clave, son más fáciles de mantener y proporcionan un buen desempeño al procesar un gran número de registros. Se desempeñan bien en búsquedas secuenciales pero mal en búsquedas con claves.
  - Archivos hash: una estructura de archivos especializada que soporta la búsqueda por medio de una clave, transforman el valor de una clave en una dirección para proporcionar un acceso rápido. Se desempeñan bien en búsquedas con claves pero mal en búsquedas secuenciales.
  - Archivos Btree: estructura de archivos soportada por la mayoría de los DBMS que proporciona un buen desempeño tanto en búsquedas con claves como secuenciales. Un archivo Btree es un árbol multiforme, balanceado.
  - Archivos B+Tree: la variante más popular de un Btree. En un B+Tree, todas las claves se encuentran almacenadas de forma redundante en los nodos hoja. El B+Tree proporciona un mejor desempeño en las búsquedas secuenciales y de rangos.

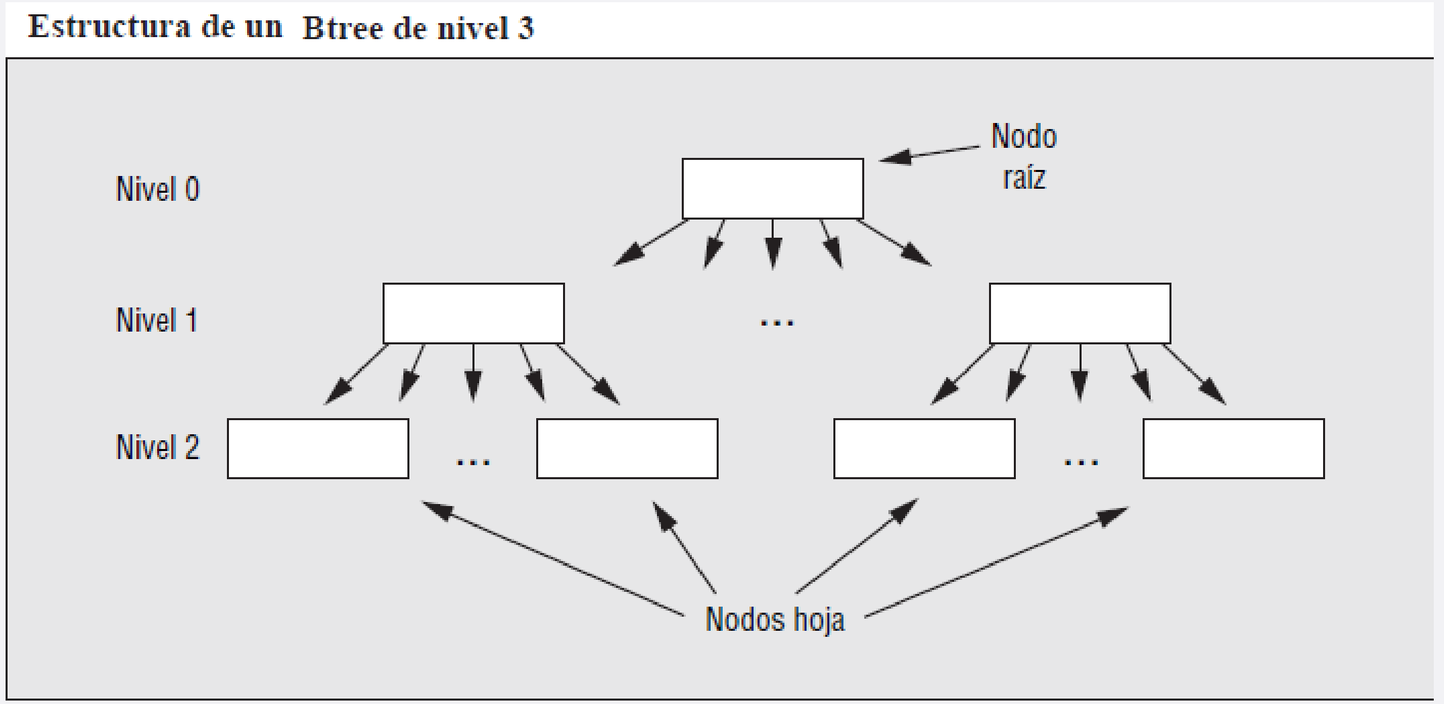
Diseño físico de las bases de datos: Generalidades (cont.)

- Características de un archivo Btree: es un tipo especial de árbol, con una estructura en la cual cada nodo tiene solo padre, a excepción del nodo raíz o nodo superior.
- Características de la estructura Btree:
  - Balanceado: todos los nodos hoja (nodos que no tienen hijos) residen en el mismo nivel del árbol.
  - Tupido (bushy): el número de ramas de un nodo es grande. Multiforme, con más de dos, es un sinónimo de arbusto. El ancho (número de flechas a partir de un nodo) y la altura (número de nodos entre los nodos raíz y hoja) están inversamente relacionados: mientras aumente el ancho, disminuye la altura. El Btree ideal es amplio (de tipo arbusto) pero pequeño (pocos niveles).
  - Orientado a bloques (*Block-Oriented*): cada nodo de un Btree es un bloque o un registro físico. Para buscar en un Btree, se comienza en la raíz y se sigue una ruta hasta el nodo hoja que contenga los datos que le interesan. La altura de un Btree es importante porque determina el número de accesos a registros físicos durante la búsqueda.
  - Dinámico: la forma de un Btree cambia mientras se insertan y borran registros lógicos. Nunca es necesario hacer una reorganización periódica para un Btree.
  - Ubicuo: el Btree es una estructura de archivos ampliamente implementada y usada.

Diseño físico de las bases de datos: Generalidades (cont.)

- Contenido de un nodo:
  - Cada nodo está formado por pares con un valor clave y un puntero (dirección física del registro), ordenados por el valor de la clave.
  - El puntero identifica el registro físico que contiene el registro lógico con el valor de la clave. Los otros datos pueden almacenarse en registros físicos separados, o en los nodos hoja.
  - Cada nodo, excepto el raíz, debe estar lleno por lo menos hasta la mitad.
  - El tamaño del registro físico es de 1.024 bytes, el tamaño de la clave es de 4 bytes y el tamaño del puntero de 4 bytes.
  - La máxima capacidad de un nodo es de 128 pares <clave, puntero>
  - El tamaño de la clave determina el número de ramas.
  - Generalmente no son buenos cuando se usan tamaños de claves grandes, ya que se tienen menos ramas por cada nodo resultando Btrees más altos y menos eficientes.

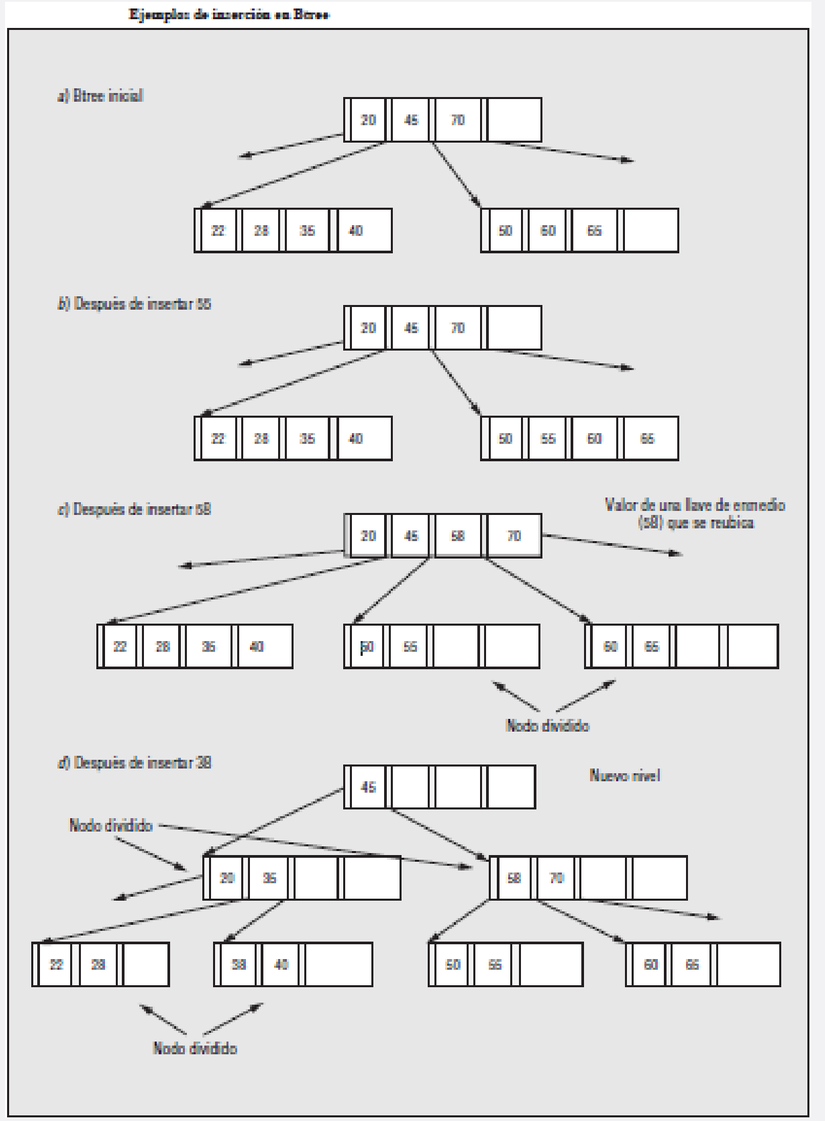
Diseño físico de las bases de datos: Generalidades (cont.)



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.10, pp. 260), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

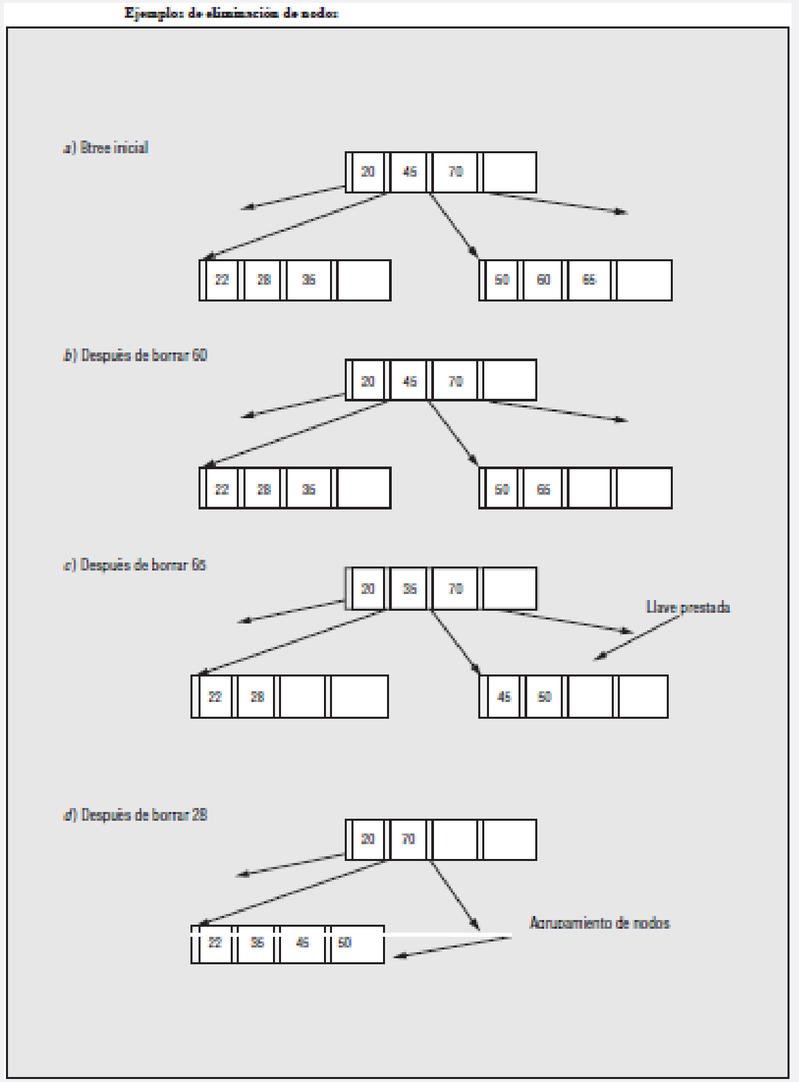
- División e integración de nodos:
  - Las inserciones suceden al colocar una nueva clave en un nodo que no esté lleno o bien dividiendo nodos.
  - Puede suceder que al agregar un valor nuevo, el nodo se divida en dos nodos y se coloque un valor de la clave en el nodo raíz, el nodo se dividirá en dos niveles.
  - Si ambos nodos están llenos, cuando ocurre una división en la raíz, el árbol crece otro nivel.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.11, pp. 261), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- División e integración de nodos:
  - Las eliminaciones se manejan quitando la clave eliminada de un nodo y reparando la estructura en caso de ser necesario.
  - Si el nodo aún se encuentra parcialmente lleno, ninguna acción adicional ocurrirá.
  - Si el nodo se encuentra a menos de la mitad, se modifica la estructura.
  - Si un nodo vecino contiene más de la mitad de su capacidad, se puede pedir prestada una clave.
  - Si no se puede obtener una clave, se deben juntar los nodos.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.11, pp. 262), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

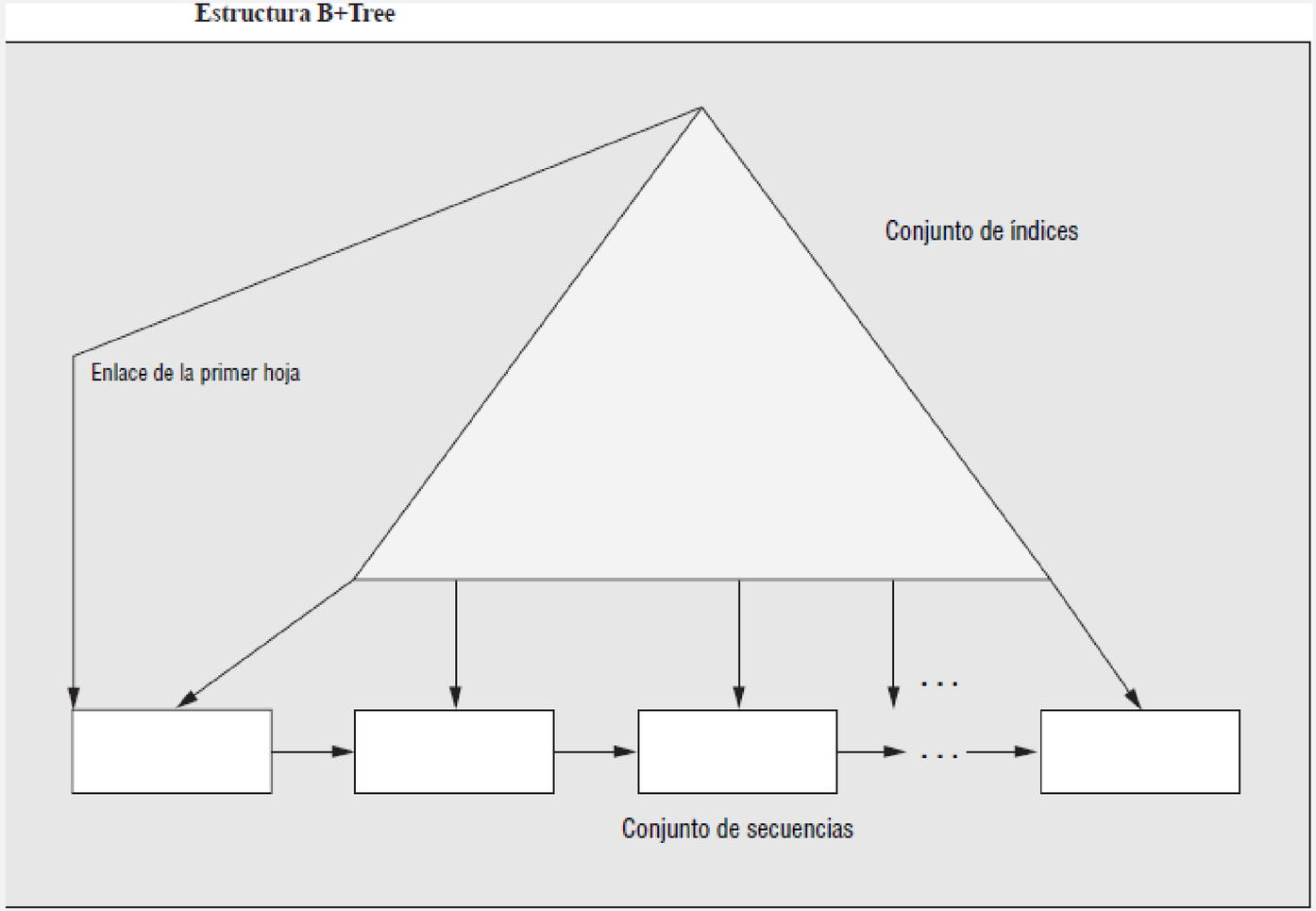
- Costos de las operaciones:
  - La altura de un Btree es más pequeña para una tabla grande cuando el factor de ramificación es grande.
  - En las operaciones Btree la altura domina el número de accesos a registros físicos.
  - El costo en términos de accesos a registros físicos para encontrar una clave es menor que o igual a la altura.
  - Si los datos de la fila no están almacenados en el árbol, se requiere de otro acceso a un registro físico para obtener los datos de la fila después de encontrar la clave.
  - El costo de insertar alguna clave incluye el costo para localizar la clave más cercana, más el costo de modificar los nodos.
  - En el mejor de los casos, el costo adicional es un acceso a un registro físico para modificar el registro del índice y un acceso a un registro físico para escribir los datos de la fila.
  - El peor de los casos ocurre cuando se agrega un nivel nuevo al árbol, teniendo en cuenta que en este último escenario aún domina la altura del árbol.

Diseño físico de las bases de datos: Generalidades (cont.)

- Costos de las operaciones (cont.):
  - Las búsquedas secuenciales pueden ser un problema con los Btrees.
  - Para realizar una búsqueda de un rango, el procedimiento de búsqueda debe viajar hacia arriba y hacia abajo del árbol.
  - Este procedimiento presenta problemas con la conservación de registros físicos en memoria.
  - Los sistemas operativos pueden reemplazar los registros físicos si no fueron accesados recientemente.
  - Debido a que suele pasar cierto tiempo antes de que se acceda nuevamente a un nodo padre, el sistema operativo puede reemplazarlo con otro registro físico en caso de que la memoria principal se llene.
  - Por ende, tal vez sea necesario otro acceso a un registro físico cuando se acceda nuevamente al nodo padre.
  - Para asegurar que no se reemplacen los registros físicos, usualmente se implementa la variación del B+Tree.

Diseño físico de las bases de datos: Generalidades (cont.)

- B+Tree:



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.14, pp. 264), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- B+Tree:
  - El triángulo (conjunto de índices) representa un índice normal de un Btree.
  - La parte inferior (conjunto de secuencias) contiene los nodos hoja.
  - Todas las claves se ubican en los nodos hoja incluso cuando una clave aparece en el conjunto de índices.
  - Los nodos hoja están conectados para que las búsquedas secuenciales no necesiten moverse hacia arriba del árbol.
  - Una vez que se encuentra la clave inicial, el proceso de búsqueda accede únicamente a los nodos del conjunto de la secuencia.

Diseño físico de las bases de datos: Generalidades (cont.)

- Coincidencia de índices:
  - Se refiere a la determinación de si se puede usar un índice en una consulta.
  - Se puede usar un Btree para almacenar todos los datos en los nodos (estructura de archivos primaria) o solo los punteros a los registros de datos (estructura de archivos secundaria o índice).
  - Un Btree es especialmente versátil como índice, ya que se puede usar para almacenar diversas consultas.
  - En una cláusula WHERE cuando una condición hace referencia a una columna indexada, el DBMS debe determinar si es posible usar el índice.
  - Para índices de columna única, un índice coincide con una condición si la columna aparece sola, sin funciones u operadores, y el operador de comparación coincide con uno de los siguientes elementos:
    - =, >, <, >=, <= (pero no < >)
    - BETWEEN
    - IS NULL
    - IN <lista de valores constantes>
    - LIKE ‘Patrón’ en el cual patrón no contiene ningún metacarácter (%, _) como la primera parte del patrón.

Diseño físico de las bases de datos: Generalidades (cont.)

- Coincidencia de índices (cont.):
  - Para los índices compuestos que involucren más de una columna, las reglas de coincidencia son más complejas y restrictivas.
  - Los índices compuestos están ordenados de la columna más significativa (primera columna del índice) a la columna menos significativa (última columna del índice).
  - Un índice compuesto coincide con las condiciones de acuerdo con las siguientes reglas:
    - La primera columna del índice debe tener una condición de coincidencia.
    - Las columnas coinciden de izquierda (más significativo) a derecha (menos significativo).
    - La coincidencia se detiene cuando la siguiente columna del índice no coincide.
    - Por lo menos una condición BETWEEN coincide.
    - Ninguna otra condición coincide después de la condición BETWEEN.
    - Por lo menos una condición IN coincide con una columna índice.
    - Las coincidencias se detienen después de la siguiente condición.
    - La segunda condición no puede ser IN o BETWEEN.

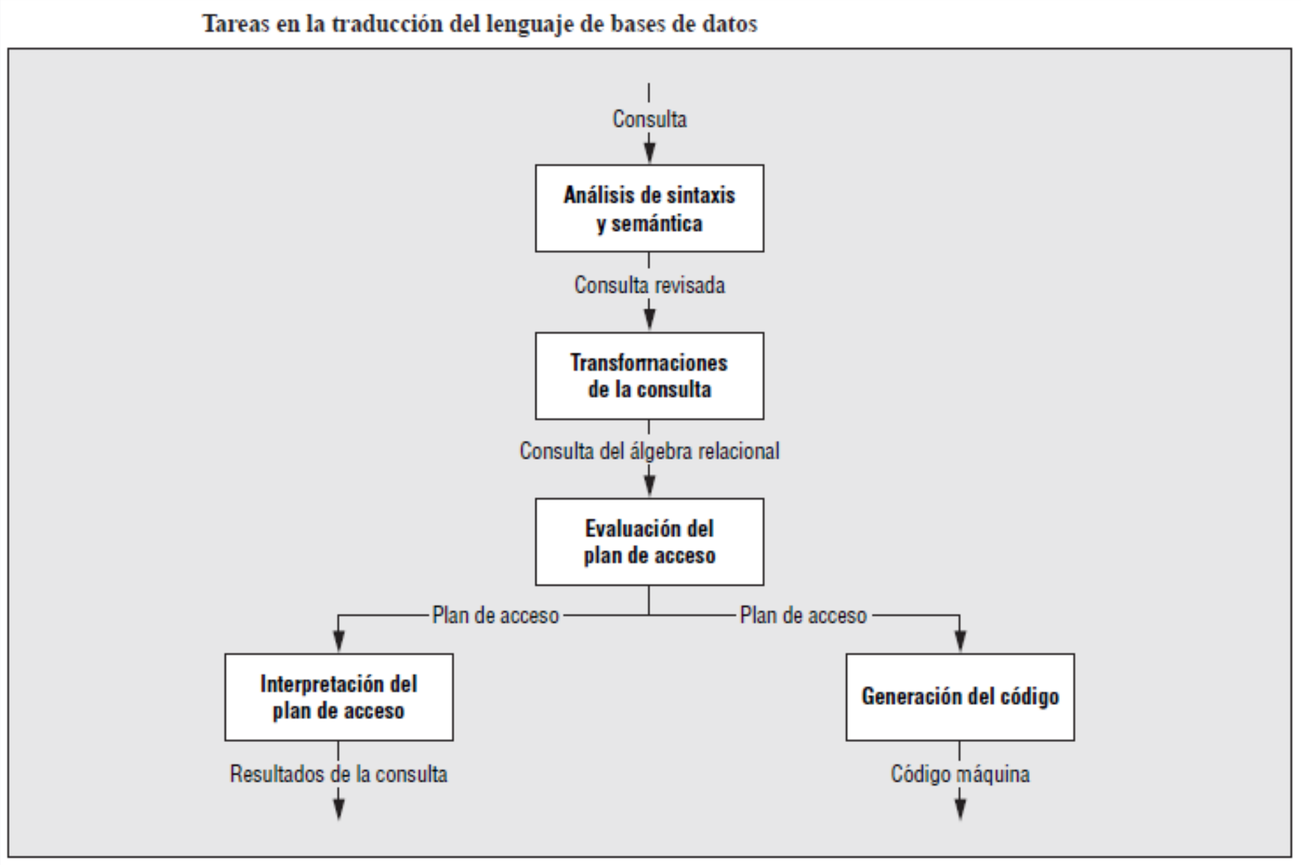
Diseño físico de las bases de datos: Generalidades (cont.)

- Coincidencia de índices (cont.):
  - Cuando se utiliza un índice compuesto, las condiciones pueden estar en cualquier orden.
  - Los índices compuestos se deben usar con precaución debido a las reglas de restricción de coincidencias.
  - Generalmente, es mejor crear índices sobre las columnas individuales, ya que la mayoría de los DBMS pueden combinar los resultados de varios índices cuando se responde a una consulta.

- Índices bitmap:
  - Estructura secundaria de archivos consistente en un valor de columna y un bitmap.
  - Un bitmap contiene una posición de bit para cada fila de la tabla referenciada.
  - Un índice de columna bitmap hace referencia a las filas que contienen el valor de la columna.
  - Un índice bitmap de enlace hace referencia a las filas de una tabla hija que se une con filas de la tabla madre contenidas en la columna.
  - Los índices bitmap funcionan correctamente para columnas estables con algunos valores típicos de tablas en un almacén de datos.

Optimización de consultas

- El componente de optimización de consultas asume la responsabilidad de escoger la mejor implementación de las consultas sobre la base de datos física.
- En algunas ocasiones se pueden mejorar esas decisiones de optimización comprendiendo los principios del proceso de optimización.
- Tareas de traducción: El componente de optimización de consultas traduce la consulta SQL en cuatro fases.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.17, pp. 269), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Optimización de consultas (cont.)

- Fase 1: Sintaxis y análisis de semántica
  - Analiza una consulta para encontrar errores de sintaxis y semántica simple.
  - Los errores de sintaxis involucran el mal uso de palabras clave, por ejemplo la palabra reservada FROM.
  - Los errores de semántica incluyen el mal uso de columnas y tablas.
  - Para encontrar errores de semántica, el DBMS usa las definiciones de tablas, columnas y relaciones, tal como se almacenan en el diccionario de datos.
  - El compilador del lenguaje de datos puede detectar únicamente errores simples de semántica que involucren tipos de datos incompatibles.
  - Por ejemplo, una condición WHERE que compare columnas con tipos de datos incompatibles, generando un error semántico.

Optimización de consultas (cont.)

- Fase 2: Transformación de consultas
  - Transforma una consulta en un formato simplificado y estandarizado.
  - Como con la optimización de los compiladores de lenguajes de programación, los traductores del lenguaje de base de datos pueden eliminar las partes redundantes de una expresión lógica. Por ejemplo, la expresión lógica (OffYear = 2006 AND OffTerm = ‘WINTER’) OR (OffYear = 2006 AND OffTerm = ‘SPRING’) se puede simplificar en OffYear = 2006 AND (OffTerm = ‘WINTER’ OR OffTerm = ‘SPRING’).
  - El formato estandarizado generalmente se basa en el álgebra relacional.
  - Las operaciones del álgebra relacional se reacomodan de tal forma que la consulta se pueda ejecutar de manera más rápida.
  - Las operaciones de restricción se combinan para que se puedan probar de manera conjunta.
  - Las operaciones de proyección y restricción se mueven para que estén antes de las operaciones de enlace (join) para eliminar las columnas y renglones innecesarios antes de las operaciones de enlace costosas.
  - Las operaciones de productos cruz se transforman en operaciones de enlace si una condición existe en la cláusula WHERE.

Optimización de consultas (cont.)

- Fase 3: Evaluación del plan de accesos
  - Genera un plan de acceso para implementar la consulta reacomodada del álgebra relacional.
  - Un plan de accesos indica la implementación de una consulta como operaciones en archivos.
  - En un plan de accesos, los nodos hoja son tablas individuales de la consulta y las flechas apuntan hacia arriba para indicar el flujo de datos.
  - Los nodos que están sobre los nodos hoja señalan las decisiones sobre el acceso a las tablas individuales.
  - Los índices Btree se usan para acceder a las tablas individuales.
  - Las estructuras de archivos Btree proporcionan la clasificación requerida por algoritmos de MERGE JOIN.
  - El componente de optimización de consultas evalúa un gran número de planes de acceso.
  - Los planes de acceso varían debido al orden de los JOINS, estructuras de archivo y algoritmos de JOIN.

Optimización de consultas (cont.)

- Fase 3: Evaluación del plan de accesos (cont.)
  - Para las estructuras de archivos, algunos componentes de optimización pueden considerar operaciones de conjuntos (intersección para las condiciones conectadas por un AND y las condiciones de enlace conectadas por un OR) para combinar los resultados de múltiples índices sobre la misma tabla.
  - El componente de optimización de consultas puede evaluar muchos más planes de acceso que los que un programador de bases de datos experimentado pudiese considerar.
  - La evaluación de los planes de acceso puede involucrar una significativa cantidad de tiempo cuando la consulta contiene más de cuatro tablas.
  - La mayoría de los componentes de optimización utilizan un pequeño conjunto de algoritmos de *join*.
  - Para cada operación de *join* en una consulta, el componente de optimización considera cada algoritmo de *join* soportado.
  - Para los *join* anidados y los algoritmos híbridos, el componente de optimización también debe seleccionar las tablas externa e interna.
  - Todos los algoritmos, a excepción del enlace de estrella (*star join*), involucran dos tablas al mismo tiempo.

Optimización de consultas (cont.)

- Fase 3: Evaluación del plan de accesos (cont.)
  - El enlace estrella puede combinar cualquier número de tablas que coincidan con el patrón de estrella (una tabla hija rodeada por tablas madre con relaciones 1-M).
  - El algoritmo de ciclos anidados se puede usar con cualquier operación de enlace, no sólo con una operación EQUI-JOIN.
  - El componente de optimización de consultas usa fórmulas de costos para evaluar los planes de acceso.
  - Cada operación en un plan de acceso tiene una fórmula de costo correspondiente que estima los accesos de registros físicos y operaciones del CPU.
  - Las fórmulas de costos usan los perfiles de las tablas para estimar el número de filas de un resultado. Por ejemplo, el número de filas que se generan de una condición WHERE se puede estimar usando los datos de distribución, tales como un histograma.
  - El componente de optimización de consultas selecciona el plan de acceso con el menor costo.

Diseño físico de las bases de datos: Generalidades

Resumen de algoritmos de JOIN comunes:

Algoritmo	Requerimientos	Cuándo usarlo
Ciclos anidados	Seleccione la tabla externa e interna; se puede usar para todos los *joins.*	Apropiado cuando existen pocas filas en la tabla externa o cuando todas las páginas de la tabla interna caben en la memoria. Un índice en la llave foránea de la columna de *join* permite un uso eficiente del algoritmo de ciclos anidados cuando existen condiciones de restricción en la tabla madre
Mezcla ordenada (*sort merge*)	Ambas tablas deben estar ordenadas (o usar un índice) en las columnas de *join*; solo se usa para los *equi-joins.*	Apropiado si el costo del ordenamiento es pequeño o si existe un índice de *joins agrupados.*
Enlace híbrido (*hash join*)	Archivo hash interno construido para ambas tablas; solo se usa para los *equi-joins.*	Un enlace *hash* es mejor que un *sort merge* cuando las tablas no están ordenadas o no existen índices.
Enlace de estrella (*star join*)	Enlace de múltiples tablas en el cual existe una tabla hija relacionada con múltiples tablas madre en relaciones 1-M; se requiere un índice de *join* bitmap en cada tabla madre; solo se usa para *equi-joins.*	Es el mejor algoritmo de *join* para las tablas que coinciden con el patrón de estrella con índices de *join* de bitmap en especial cuando existen condiciones altamente selectivas en las tablas madre; ampliamente usado para optimizar las consultas de una *data warehouse*.

Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.8, pp. 271), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Optimización de consultas (cont.)

- Fase 4: Ejecución del plan de acceso
  - La última fase ejecuta el plan de acceso seleccionado.
  - El componente de optimización de consultas genera el código máquina o interpreta el plan de acceso.
  - La ejecución del código máquina genera una respuesta más rápida que la interpretación del plan de acceso.
  - La mayoría de los DBMS interpretan los planes de acceso debido a la amplia variedad de *hardware* soportado.
  - La diferencia en el desempeño entre la interpretación y la ejecución del código máquina generalmente no es significativa para la mayoría de los usuarios.

Resumen de entradas, salidas y entorno del diseño físico de bases de datos

Elemento	Descripción
Entradas	
Perfil de tablas	Estadísticas para cada tabla, como número de filas y de columnas de valores únicos.
Perfil de aplicación	Estadísticas para cada formulario, reporte y consulta, tales como accesos/actualizaciones y la frecuencia de los accesos/actualizaciones.
Salidas	
Estructuras de archivos	Métodos de organización de registros físicos para cada tabla.
Colocación de datos	Criterios para acomodar los registros físicos de forma cercana.
Formateo de datos	Uso de la comprensión y de los datos derivados.
Desnormalización	Combinación de tablas separadas en una sola tabla
Conocimiento del entorno	
Estructuras de archivos	Características como las operaciones respaldadas y fórmulas de costo.
Optimización de consultas	Decisiones de accesos hechas por el componente de optimización para cada una de las consultas.

Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.1, pp. 252), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Ejemplo práctico. Tablas a considerar

Ejemplo de tabla de inscripción *Enrollment*

OfferNo	StdSSN	EnrGrade
1234	123-45-6789	3.3
1234	234-56-7890	3.5
4321	123-45-6789	3.5
4321	124-56-7890	3.2

Ejemplo de tabla de cursos ofrecidos *Offering*

OfferNo	CourseNo	OffTerm	OffYear	OffLocation	OffTime	FacSSN	OffDays
1111	IS320	SUMMER	2006	BLM302	10:30 AM	-	MW
1234	IS320	FALL	2005	BLM302	10:30 AM	098-76-5432	MW
2222	IS460	SUMMER	2005	BLM412	1:30 PM	-	TTH
3333	IS320	SPRING	2006	BLM214	8:30 AM	098-76-5432	MW
4321	IS320	FALL	2005	BLM214	3:30 PM	098-76-5432	TTH
4444	IS320	SPRING	2006	BLM302	3:30 PM	543-21-0987	TTH
5678	IS480	SPRING	2006	BLM302	10:30 AM	987-65-4321	MW
5679	IS480	SPRING	2006	BLM412	3:30 PM	876-54-3210	TTH
9876	IS460	SPRING	2006	BLM307	1:30 PM	654-32-1098	TTH

Nota: Tablas adaptadas de *Administración de base de datos diseño y desarrollo de aplicaciones* (tab. 3.3, pp. 48 y tab. 3.4, pp. 48), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Ejemplo práctico. Tablas a considerar (cont.)

Ejemplo de tabla de inscripción *Faculty*

FacSSN	FacFistName	FacLastName	FacCity	FacState	FacDept	FacRank	FacSalary	FacSupervisor	FacHireDate	FacZipCode
098-76-5432	LEONARD	VINCE	SEATTLE	WA	MS	ASST	$35,000	654-32-1098	01-Apr-95	98111-9921
543-21-0987	VICTORIA	EMMANUEL	BOTHELL	WA	MS	PROF	$120,000	-	01-Apr-96	98111-2242
654-32-1098	LEONARD	FIBON	SEATTLE	WA	MS	ASSC	$70,000	543-21-0987	01-Apr-95	98121-0094
765-43-2109	NICKI	MACON	BELLEVUE	WA	FIN	PROF	$65,000	-	01-Apr-97	98015-9945
876-54-3210	CRISTOPHER	COLAN	SEATTLE	WA	MS	ASST	$40,000	654-32-1098	01-Apr-99	98114-1332
987-65-4321	JULIA	MILLS	SEATTLE	WA	FIN	ASSC	$75,000	765-43-2109	01-Apr-00	98114-9954

Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 3.7, pp. 52), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Ejemplo práctico. SQL a considerar

Enlace de tres tablas

```
SELECT FacName, CourseNo, Enrollment.OfferNo, EnrGrade
        FROM Enrollment, Offering, Faculty
        WHERE CourseNo LIKE ‘IS%’ AND OffYear = 2005
                AND OffTerm = ‘FALL’
                AND Enrollment.OfferNo = Offering.OfferNo
                AND Faculty.FacSSN = Offering.FacSSN
```

Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.1, pp. 268), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

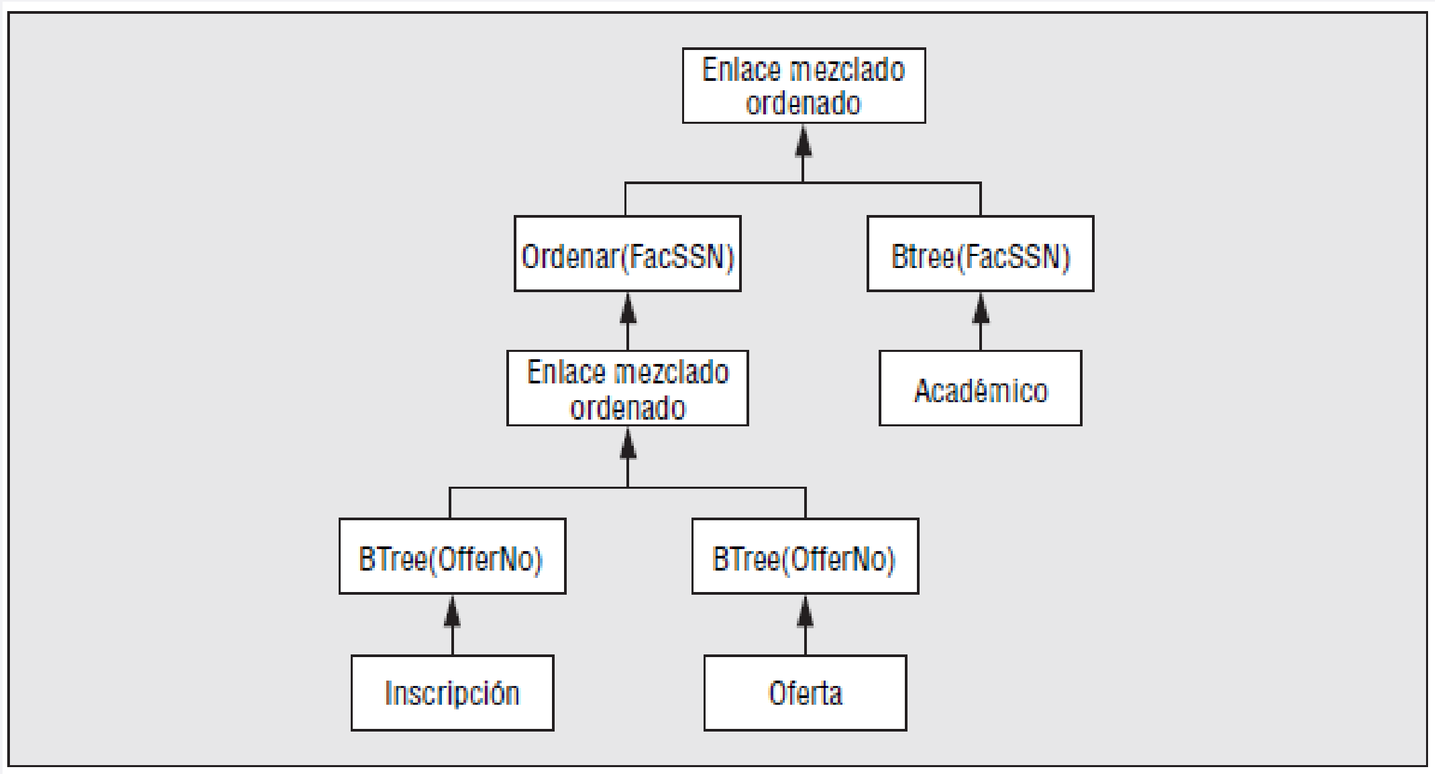
Ejemplo práctico

Plan de acceso 1:

- Los nodos hoja son tablas individuales de la consulta y las flechas apuntan hacia arriba para indicar el flujo de datos.
- Los nodos que están sobre los nodos hoja señalan las decisiones sobre el acceso a las tablas individuales.
- Los índices Btree se usan para acceder a las tablas individuales.
- El primer enlace combina las tablas Enrollment y Offering.
- Las estructuras de archivos Btree proporcionan la clasificación requerida por el algoritmo de merge join.
- El segundo enlace (*join*) combina el resultado del primer *join* con la tabla Faculty.
- El resultado intermedio debe estar ordenado en FacSSN antes de que se pueda usar el algoritmo de *merge join*.

Ejemplo práctico

Gráfico plan de acceso 1:

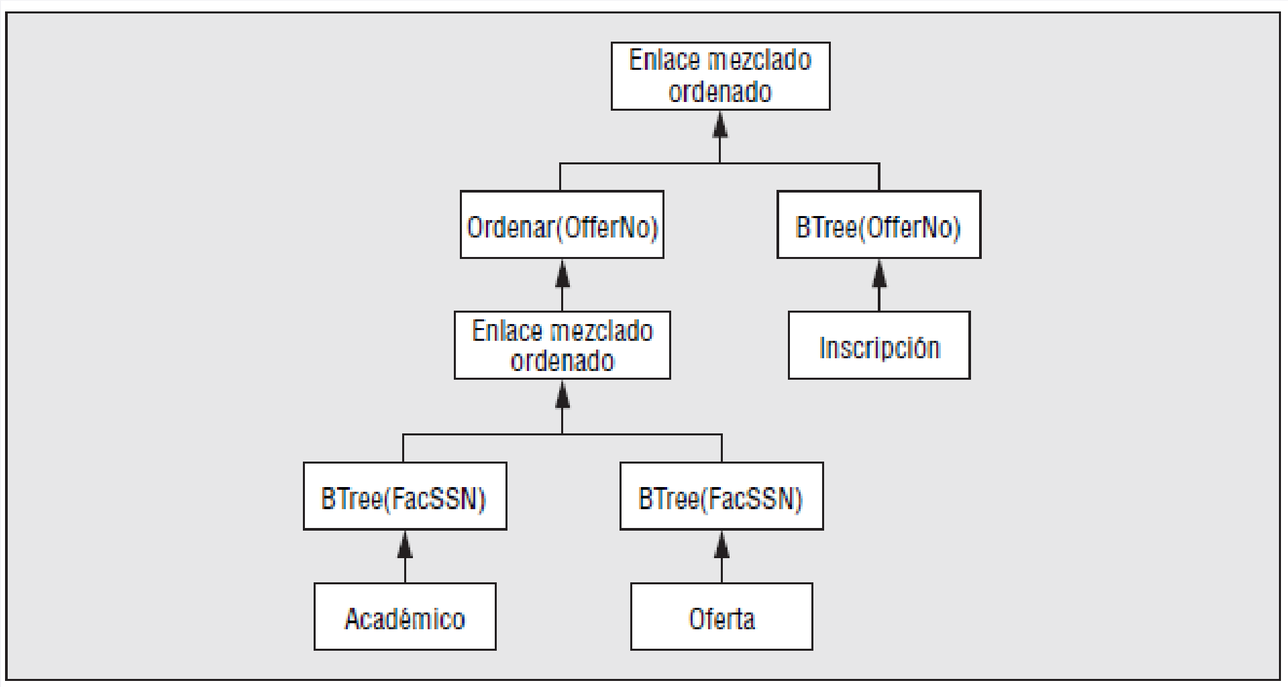


Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.18, pp. 270), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Ejemplo práctico

Plan de acceso 2:

- Variación del plan de acceso 1 en el que el orden del *join* está modificado. Los planes de acceso varían debido al orden de los *joins*, estructuras de archivo y algoritmos de *join*.1



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.19, pp. 270), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.