Un poco más de R-Trees y bitmaps

R-Trees:

Costo asintótico:

Se trabajan de forma parecida a los B-Tree. Sin embargo tienen el mismo costo asintótico. Veamos:

Sean a, b son las "dimensiones" del nodo R, sea f_n el "tamaño" del nodo de profundidad *n* y sea f_0 el tamaño del conjunto inicial de datos. Entonces el tamaño de cada nodo:

$$f_1 = \frac{f_0}{ab}, f_n = \frac{f_{n-1}}{ab} \Rightarrow f_n = \left(\frac{1}{ab}\right)^n f_0$$

Se entiende que la llegada a una hoja ocurre cuando $f_n = 1$. Entonces:

$$f_n = 1 = \left(\frac{1}{ab}\right)^n f_0 \xrightarrow{\log} \ln 1 = n \ln\left(\frac{1}{ab}\right) + \ln f_0$$

Si decimos que el tamaño del bucket es $B = ab$, entonces:

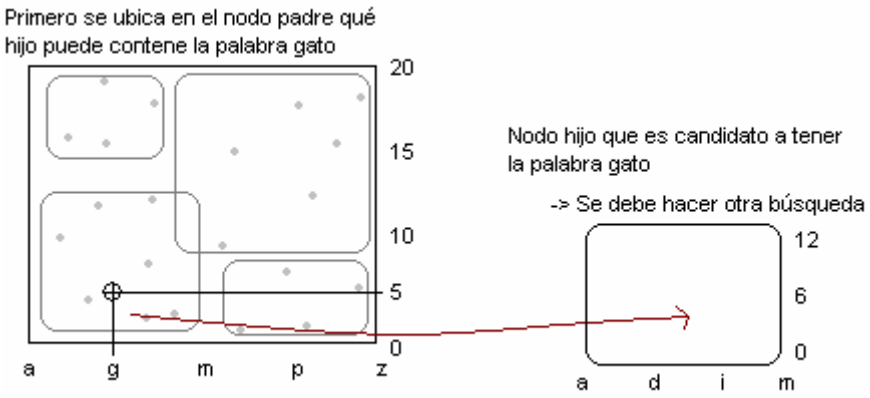
$$0 = n \ln\left(\frac{1}{B}\right) + \ln f_0 \Rightarrow n \ln B = \ln f_0 \Rightarrow n = \frac{\ln f_0}{\ln B} = \log_B f_0$$

En otras palabras, el árbol R tiene el mismo orden asintótico que el árbol B.

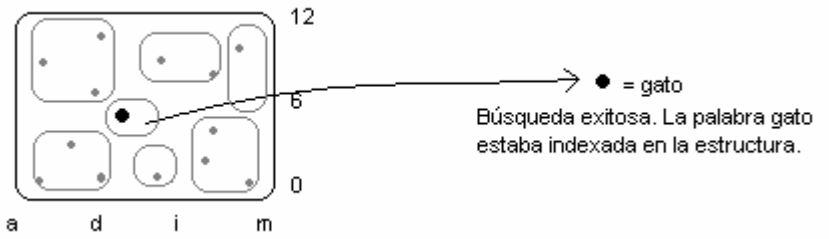
Operaciones en R-Tree:

Ocurren de manera análoga a los árboles B. Veamos en un ejemplo el agrupamiento de la información:

Ejemplo: Supongamos que tenemos indexado un conjunto de palabras en un árbol R. Estas palabras están agrupadas según dos propiedades: *Largo de palabra* y *orden alfabético*. Con esto busquemos la palabra *gato*. Esta palabra tiene 4 letras y empieza con g:



Dentro del nodo hijo hay que volver a buscar un hijo con la palabra gato. El proceso se repite hasta llegar a una hoja. Supongamos que el hijo anterior nos lleva a una hoja:



Si la hoja nos hubiese llevado a *geto*, entonces no la palabra *gato* no habría estado indexada en la base de datos.

Ahora, en lo que respecta a bases de datos, lo válido es suponer que la hoja es un puntero directo a un bucket de alguna tabla indexada y luego sería tiempo de procesador el revisar el bucket y verificar que la palabra *gato* estaba efectivamente indexada.

Bitmaps:

Costos:

Como su nombre lo dice, los bitmaps son mapas de bits. La estructura de una bitmap es un arreglo ordenado cuyos bits se corresponden ordenadamente a una fila de alguna tabla. Cada bit del bitmap es un sí (1) o un no (0) respecto de alguna propiedad de una tabla. A modo de ejemplo, si se quisiera almacenar la propiedad "ser top ten del tenis" entonces se hace una consulta que entregue las filas que correspondan a los top ten del tenis. Las filas devueltas se almacenarían como unos en el bitmap y el resto como ceros. Ahora si tenemos el bitmap de los top ten y el bitmap de cuáles jugadores son europeos, entonces la operación [BITMAP TOP TEN] AND [BITMAP EUROPEOS] indica los jugadores europeos que están en el top ten.

Como se podrá apreciar, el costo de hacer la selección es casi costo de procesador. La lectura es muy rápida, lo que hace la estructura muy eficiente. A modo de ejemplo, supongamos que tenemos una tabla con 10 millones de filas. El espacio ocupado por cada bitmap es:

$$\frac{10^7}{8 \cdot 1024} kb = \frac{1.25 \times 10^6}{1024} kb < 1.25 \times 10^3 kb = 1250 kb \approx 1.2 mb$$

Lo que es muy poco espacio comparado con las diez millones de filas. Supongamos que cada fila ocupa 0.1 *kb* de espacio. Entonces la tabla ocupa alrededor de 1 millón de *kb*, o sea, cerca de 1 *gb*. Son muchos órdenes de magnitud de diferencia.

Pero el problema de los bitmaps es que hay que recalcularlos completamente luego de cada modificación. Por ejemplo, usemos el top ten. No hay forma de decir, con esa información, si una modificación en el puntaje de los tenistas saca a uno u otro. Entonces el cálculo se hace de nuevo.

Para el caso de 10 millones de filas a 0.1 *kb* cada una, procesar 1 *gb* es algo muy costoso. Por eso es que los bitmaps se aplican sobre bases de datos estáticas y de consultas específicas y recurrentes.

Operaciones en bitmaps:

Ejemplo: Tenemos dos tablas, autos y botes. A1 y B1 son los bitmaps de los autos y botes Yamaha. A2 y B2 son los bitmaps de los autos y botes posteriores al año 2000. Entonces:

- El modelo de los autos Yamaha posteriores al 2000: Son la proyección sobre el campo *modelo* de las filas entregadas por el bitmap resultante A1 AND A2.
- El motor de los autos y botes Yamaha: Son la proyección sobre el campo *motor* de las filas de la tabla auto asociadas a A1 y de la tabla botes asociadas a B1.
- Todos los botes no Yamaha fabricados hasta el 2000, inclusive: Son la proyección sobre botes de NOT B1 AND NOT B2.
- Todos los autos fabricados luego del 2003: Es la selección sobre los autos que además cumplen con A2. Y es que también fueron fabricados después del 2000.

Estas propiedades son las que hacen deseable la existencia de bitmaps para ciertas bases de datos. Sin embargo esa eficiencia en selección tiene un alto costo en construcción.