**Estudiante:** Juan José Mahecha Villamil

**Código:** 100012373

**Programa:** Ingeniería de software 5 semestre promoción 11

**Materia:** Estructura de Datos

**Documentación del Proyecto:** Sistema de Inventario de Celulares

Para este trabajo de estructura de datos se ha usado MySQL, para hacer búsquedas básicas pero rápidas de datos y optimizar el acceso, búsqueda y almacenamiento de la información. En específico los árboles de derivación B y B+. En MySQL, estos árboles se utilizan en los índices para acelerar y eficientizar las búsquedas.

**¿Por qué árboles?**

La opción más factible para hacer búsquedas precisas, inserciones y eliminaciones en tiempos reducidos en bases de datos grandes son los árboles de derivación donde la eficiencia es crucial, esta estructura de datos es la mejor opción para lograr tiempos de respuesta rápidos, especialmente en este proyecto que es un inventario de celulares.

El árbol de búsqueda implementado fue el modelo BST (Binary Search Tree), cuenta con datos en la que cada nodo tiene hasta 2 hijos, los valores en los nodos van siguiendo una propiedad que les facilita la búsqueda, todos los nodos en el subárbol derecho tienen valores mayores, izquierdo menores y está el axioma que es el papá de todos, el nodo principal.

¿Por qué el modelo de árbol BST?: Para este CRUD de inventario de celulares usé el BST principalmente para optimizar la búsqueda de registros en la base de datos, fácil uso, simplicidad, con una comodidad para el usuario que la usé.

**¿Cómo funciona?:**

En mi proyecto, cuando un usuario quiere buscar un celular, lo puede hacer por medio de su nombre, el sistema que diseñé lleva a cabo una consulta SQL que aprovecha el índice “CREATE INDEX nombre ON inventario\_celulares”, se creó una columna en la parte principal llamada "nombre", utiliza el árbol B o B+ para encontrar eficientemente el nombre en la base de datos. Este árbol asegura que, independientemente de la cantidad de registros que tenga, la búsqueda se realice de manera veloz, que es precisamente lo que necesitamos, velocidad de búsqueda en datos almacenados grandes.  
  
Ejemplo práctico del árbol BST en la aplicación:  
Supongamos que Javier está usando la aplicación de inventarios de celular y necesita encontrar cuanta disponibilidad existe del “iPhone 12”, Javier encuentra el campo para buscar celular e ingresa “iPhone 12” y presiona el botón “Buscar”.

**¿Qué hace ahora?:**

En lugar de buscar en todos los registros uno por uno (lo que sería una búsqueda lineal), el sistema utiliza el árbol BST que se implementó. Comienza en el nodo raíz, que podría ser un celular cuyo nombre esté en el medio del rango alfabético de todos los celulares en el inventario.

Redmi note 11

Google Píxel 4

Cubot P30

Huawei P30"

Samsung Galaxy S20

iPhone 12

El celular "iPhone12" estaría en el medio del rango alfabético de nombres. Por lo tanto, el nodo raíz del árbol de búsqueda binaria es "iPhone".

Para lograr este tipo de búsqueda, se utiliza una consulta SQL que incluye un operador como "%" para buscar cualquier coincidencia parcial en la columna "nombre" de la base de datos. Por ejemplo:

SELECT \* FROM inventario\_celulares WHERE nombre LIKE '%iphone%';

**¿Cómo continua?**

El sistema compara "iPhone 12" con el nombre del celular en el nodo raíz. Dado que "iPhone 12" es alfabéticamente menor que el nombre en el nodo raíz, se mueve al subárbol izquierdo, los cuales son los celulares cuyos nombres comienzan con letras antes de "i" (como "Apple iPhone 11" y "Google Pixel 4") serían considerados en la búsqueda. "iPhone 12" no estaría en ese subárbol izquierdo, ya que su nombre comienza con "i.".

Si buscas "iPhone 12," la comparación te llevaría al subárbol derecho, y solo se considerarían los celulares cuyos nombres comienzan con letras después de "i" en el alfabeto. En este caso, ninguno de los celulares en la lista cumpliría con ese criterio, ya que todos tienen nombres que comienzan con letras antes de "i."

Ahora, el sistema compara "iPhone 12" con el nombre en el nodo raíz del subárbol izquierdo. Si no coincide, se mueve nuevamente a la izquierda o a la derecha, según sea necesario, siguiendo la propiedad del árbol BST.

El proceso continúa hasta que el sistema encuentre el nodo que contiene el celular con el nombre "iPhone 12" o determine que no existe en el inventario.

**Ejemplo gráfico:**

Diagrama

Descripción generada automáticamente  
La raíz es "Samsung Galaxy S21."

Todos los nodos a la izquierda de un nodo contienen valores alfabéticamente anteriores, y todos los nodos a la derecha contienen valores alfabéticamente posteriores.

Ahora, supongamos que estás buscando "iPhone 12." La comparación comienza en la raíz:

"iPhone 12" vs. "Samsung Galaxy S21": "iPhone 12" es alfabéticamente posterior, por lo que nos movemos al subárbol derecho.

"iPhone 12" vs. "Huawei P30 Pro": "iPhone 12" es alfabéticamente posterior, por lo que nos movemos al subárbol derecho.

"iPhone 12" vs. "Apple iPhone 11": "iPhone 12" es alfabéticamente posterior, por lo que nos movemos al subárbol derecho.

"iPhone 12" vs. "Xiaomi Mi 10": "iPhone 12" es alfabéticamente anterior, por lo que nos movemos al subárbol izquierdo.

"iPhone 12" vs. "iPhone 12": ¡Encontramos una coincidencia!

Dado que hemos encontrado una coincidencia en el nodo "iPhone 12," podemos concluir que "iPhone 12" está en la base de datos.

Soft delete o borrado lógico usando UUID para los ID

**Conclusión:**

Este enfoque es mucho más eficiente que una búsqueda lineal, especialmente cuando tienes una gran cantidad de registros en tu base de datos. El árbol BST permite reducir drásticamente el número de comparaciones necesarias para encontrar el registro deseado, lo que acelera el proceso de búsqueda.