Método de la ingeniería

Alejandro Arce

Duvan Cuero

Alexander Samaca

Universidad Icesi

Algoritmos y Estructuras de datos

Santiago de Cali

3 Noviembre del 2020

**ETAPA 1: Contexto del problema**

Después del excelente rendimiento en sus trabajos académicos previos, sus profesores de algoritmos los han recomendado a diversos profesores de la Facultad de Ingeniería, para que trabajen en sus proyectos de investigación. Su equipo ha sido contratado para una monitoría en un proyecto de investigación interna de la Universidad, como parte del Equipo VIP de Simulación[[1]](#footnote-0).

El sub-proyecto que le ha sido asignado, consiste en el desarrollo de un prototipo de software que permita gestionar eficientemente las operaciones CRUD (Create, Read, Update y Delete) sobre una base de datos de personas de nuestro continente. La población del continente americano se estima, en 2020, en poco más de mil millones de personas[[2]](#footnote-1), representando cerca del 13% del total mundial. Por tanto, usted debe simular la creación de (generar) un número similar de registros de personas, para este continente, con los siguientes datos: código (autogenerado), nombre, apellido, sexo, fecha de nacimiento, estatura, nacionalidad y fotografía.

**Definición del problema**

El sub-proyecto requiere gestionar eficientemente las operaciones CRUD en una base de datos de personas de nuestro continente de más o menos mil millones de personas.

**Especificación de requerimientos**

* **Requerimientos funcionales**

El sistema está en la capacidad de:

* Generar un registro de personas del continente americano con los siguientes datos: código (generado automáticamente), nombre, apellido, sexo, fecha de nacimiento, estatura, nacionalidad y fotografia. El programa cuenta con un campo de texto donde se pone el número de registros que se desea generar, este número de registros tiene un máximo que es de más o menos mil millones que corresponde al número de personas del continente americano. Además se debe mostrar una barra de progreso en el momento de generar los registros
* Agregar un nuevo registro de una persona con los datos : código (generado automáticamente), nombre, apellido, sexo, fecha de nacimiento, estatura, nacionalidad y fotografia. Y cuenta con una opción guardar de manera persistente
* Actualizar la información de una persona en todos los campos editables menos el código que es autogenerado.
* Buscar una persona en la base de datos dado uno de los siguientes criterios: nombre, apellido, nombre completo y código. Para el nombre y el nombre completo se despliega una lista emergente con maximo 100 busquedas que cuenten con los primeros caracteres en orden al momento de digitar.
* Eliminar un registro de una persona en la base de datos.
* De guardar todos los cambios en el mismo programa, es decir, si se hace algún cambio en la base de datos ya sea actualizar información, agregar personas o eliminar, esta base de datos se guardará aunque se cierre el programa y se vuelva abrir.

**Requerimientos no funcionales**

* Al momento de almacenar todos los registros de personas se debe utilizar árboles binarios de búsquedas autobalanceados serializados.
* Para generar los nombres completos se debe tomar los nombres de este dataset [dataset de nombres de data.world](https://data.world/alexandra/baby-names). Para los apellidos se debe usar este dataset [dataset de apellidos de data.world](https://data.world/uscensusbureau/frequently-occurring-surnames-from-the-census-2010).
* Para generar la fecha de nacimiento suponemos que la distribución de edad de toda América es igual a [esta distribución de edad de Estados Unidos](https://www.indexmundi.com/es/estados_unidos/distribucion_por_edad.html)**.**
* Para generar la estatura de los registros debe ser de manera aleatoria en un rango coherente.
* Para generar la nacionalidad de las personas debe seguir los porcentajes relativos de población que corresponde a cada país, se debe tomar [estos datos de población por países](https://www.kaggle.com/tanuprabhu/population-by-country-2020) como base para generar las nacionalidades. Se debe tener en cuenta que si existe una diferencia en el total, esta diferencia ya sea negativa o positiva se agregara al país con mayor población.
* La barra de progreso al momento de generar los registros demora más de 1 segundo se debe mostrar el tiempo que demoró cada operación.

**ETAPA 2 : *Recopilación de información:***

Para resolver este problema lo fundamental es reunir toda la información sobre operaciones CRUD y bases de datos. Como vamos a usar nuestra propia estructura de datos un árbol autobalanceado de búsqueda(BST AVL) también se definirán estos conceptos.

*CRUD*

En informática, CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

*Base De Datos*

Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta (<https://es.wikipedia.org/wiki/Base_de_datos>)

*¿Qué es un Árbol como estructura de datos?*

Es una [estructura de datos](https://es.wikipedia.org/wiki/Estructura_de_datos) de tipo [árbol](https://es.wikipedia.org/wiki/%C3%81rbol_(programaci%C3%B3n)) que permite la recuperación de información (de ahí su nombre del [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s) re**TRIE**val). La información almacenada en un trie es un conjunto de claves, donde una clave es una secuencia de símbolos pertenecientes a un alfabeto. Las claves son almacenadas en las hojas del árbol y los nodos internos son pasarelas para guiar la búsqueda. El árbol se estructura de forma que cada letra de la clave se sitúa en un nodo de forma que los hijos de un nodo representan las distintas posibilidades de símbolos diferentes que pueden continuar al símbolo representado por el nodo padre. Por tanto la búsqueda en un trie se hace de forma similar a como se hacen las búsquedas en un diccionario:

Se empieza en la raíz del árbol. Si el símbolo que estamos buscando es A entonces la búsqueda continúa en el subárbol asociado al símbolo A que cuelga de la raíz. Se sigue de forma análoga hasta llegar al nodo hoja. Entonces se compara la cadena asociada a el nodo hoja y si coincide con la cadena de búsqueda entonces la búsqueda ha terminado en éxito, si no entonces el elemento no se encuentra en el árbol.

Por eficiencia se suelen eliminar los nodos intermedios que sólo tienen un hijo, es decir, si un nodo intermedio tiene sólo un hijo con cierto carácter entonces el nodo hijo será el nodo hoja que contiene directamente la clave completa.

Es muy útil para conseguir búsquedas eficientes en repositorios de datos muy voluminosos. La forma en la que se almacena la información permite hacer búsquedas eficientes de cadenas que comparten prefijos.

(https://es.wikipedia.org/wiki/Trie)

*¿Entonces que es un Árbol Autobalanceado?*

En [ciencias de la computación](https://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n), un **árbol binario de búsqueda auto-balanceable** o **equilibrado** es un [árbol binario de búsqueda](https://es.wikipedia.org/wiki/%C3%81rbol_binario_de_b%C3%BAsqueda) que intenta mantener su altura, o el número de niveles de nodos bajo la raíz, tan pequeños como sea posible en todo momento, automáticamente. Esto es importante, ya que muchas operaciones en un árbol de búsqueda binaria tardan un tiempo proporcional a la altura del árbol, y los árboles binarios de búsqueda ordinarios pueden tomar alturas muy grandes en situaciones normales, como cuando las claves son insertadas en orden. Mantener baja la altura se consigue habitualmente realizando transformaciones en el árbol, como la [rotación de árboles](https://es.wikipedia.org/wiki/Rotaci%C3%B3n_de_%C3%A1rboles), en momentos clave.

Tiempos para varias operaciones en términos del número de nodos en el árbol n:

|  |  |
| --- | --- |
| **Operación** | **Tiempo en** [**cota superior asintótica**](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica) |
| Búsqueda | O(log n) |
| Inserción | O(log n) |
| Eliminación | O(log n) |
| Iteración en orden | O(n) |

Para algunas implementaciones estos tiempos son el peor caso, mientras que para otras están amortizados.

Estructuras de datos populares que implementan este tipo de árbol:

* [Árbol AVL](https://es.wikipedia.org/wiki/%C3%81rbol_AVL)
* [Árbol rojo-negro](https://es.wikipedia.org/wiki/%C3%81rbol_rojo-negro)

(<https://es.wikipedia.org/wiki/%C3%81rbol_binario_de_b%C3%BAsqueda_auto-balanceable>)

**Etapa 3: Búsqueda de soluciones creativas.**

1. [↑](#footnote-ref-0)
2. [↑](#footnote-ref-1)