HEA: Herramienta de Software para enseñanza de árboles.

R Bertone[1], P Thomas[2], E Nucilli[3]

Instituto de Investigación en Informática – III LIDI[4]
Facultad de Informática UNLP

Resumen. Las estructuras de datos que sirven de soporte para almacenamiento de información contenida en una Base de Datos son parte del proceso de aprendizaje de una asignatura básica de BD. Los árboles balanceados son el ejemplo de estructuras para administrar el acceso eficiente a una BD. Dentro del proceso de enseñanza-aprendizaje de la cátedra Introducción a las Bases de Datos de la Facultad de Informática de la UNLP está incluido el tema de árboles balanceados y, si bien las prácticas están orientadas a una clara comprensión del tema, no se dispone de herramientas de software que permitan analizar y comprender la lógica en la construcción de árboles de la familia de árboles B, con un suficiente nivel de abstracción y con los principios establecidos en la materia. El propósito fundamental de HEA (Herramienta de software para la enseñanza de árboles) es brindar al alumno un asistente para la ayuda en el aprendizaje de conceptos básicos de árboles B. Además, realizar un estudio interdisciplinario vinculando aspectos tecnológicos, de diseño y académicos, resulta una experiencia interesante que redundará en un producto de software educativo con alta usabilidad.

Keywords: Bases de Datos, Árboles B, Herramienta Educativa

1 pbertone@lidi.info.unlp.edu.ar Profesor Titular Dedicación Exclusiva, Facultad de Informática UNLP
2 pthomas@lidi.info.unlp.edu.ar Profesor Adjunto Dedicación Exclusiva, Facultad de Informática UNLP
3 emanuel.nucilli@gmail.com Analista de Computación, Facultad de Informática, UNLP
4 50 y 120, 2do piso, La Plata, 1900, Argentina, +54 221 4227707

1 Introducción

Los árboles B surgieron en 1972 creados por R.Bayer y E.McCreight. El problema que dio por origen a este tipo de estructura fue la necesidad acceder a la información contenida en una Base de Datos (BD) de manera eficiente, generando una estructura auxiliar que permita la recuperación, inserción y modificación rápida de datos en una tabla [1].

Existen métodos y estructuras de datos que permiten realizar una búsqueda dentro de un conjunto grande de datos en un tiempo de orden logarítmico de accesos a disco($O(\log_2 n)$). Los árboles binarios permiten recuperar la información almacenada en archivos con algoritmos que respetan ese orden. Sin embargo, presentan una gran dificultad asociada, la eficiencia solo se puede mantener si la estructura se encuentra balanceada. El problema de los árboles binarios es que se desbalancean fácilmente.

Esta problemática fue solucionada con la creación de los árboles AVL, o árboles balanceados en altura. Este tipo de estructuras tiene una construcción que respeta un precepto muy simple, la diferencia entre el camino más corto y el más largo entre un nodo terminal y la raíz no puede diferir en más de un determinado delta, siendo dicho delta el nivel de balanceo en altura del árbol. Cuando se agrega un dato en el árbol AVL y este precepto no se respeta, se activa un mecanismo que permite rebalancear el árbol. Este mecanismo es de alto costo en tiempo debido a que es necesario realizar múltiples accesos al árbol que actúa como índice a fin de mantener el balanceo deseado. Entonces, el proceso de inserción puede resultar muy lento y por ende poco viable para ser utilizado por los DBMS como estructura de almacenamiento de sus índices [2].

Otra restricción de los árboles binario radica en su propia concepción, porque es una estructura que permite que cada nodo que la compone, posea un elemento con a lo sumo dos hijos. Esta limitación cuando se implementa una estructura auxiliar que permita el acceso rápido a los datos de una tabla de una BD, resulta importante. Los árboles n-arios o multicamino representan otra alternativa de trabajo.

Se define un árbol multicamino como una estructura de datos en la cual cada nodo o registro, puede contener k elementos y k+1 hijos. Se introduce, de esta manera el concepto de orden de un árbol. El orden de un árbol es la máxima cantidad de descendientes que puede tener un nodo o registro. En un árbol binario el orden es dos, en un árbol multicamino el orden es M, siendo M la máxima cantidad de hijos que puede tener un nodo [3].

Las estructuras arbóreas multicamino representan una mejor solución para las estructuras de índices. No es menester de este trabajo discutir las bondades ni los fundamentos de los árboles multicamino, pero, a modo de ejemplo, la eficiencia de búsqueda está ligada al orden del árbol. Así, un árbol multicamino de orden M tendrá una eficiencia de $O(\log_M n)$, lo cual lo torna en una solución mejor que los árboles binarios[4].

Sin embargo, nuevamente, los árboles multicamino pueden desbalancearse. Sin bien este desbalanceo se produce de una manera mucho más gradual, no se puede garantizar que el orden de eficiencia enunciado se mantenga en todo momento. Así, en una estructura de índice organizada mediante un árbol multicamino no puede asegurar que la eficiencia de búsqueda se mantenga en $O(\log_M n)$. Nuevamente es necesario abocarse en otra estructura.

Todo este proceso de enseñanza-aprendizaje es realizado en la cátedra de Introducción a las Bases de Datos de la Facultad de Informática de la UNLP de manera gradual. Se estudian cada una de las estructuras anteriores con el suficiente nivel de detalle, lo que permite al alumno analizar las ventajas de cada estructura y como esta ventaja se ve opacada, en cada caso.

De esta manera, y permitiendo un avance gradual, se presentan a los árboles balanceados como alternativa viable para la implantación de índices que permitan el acceso rápido a la una BD, ligado con procesos de altas, bajas y modificaciones eficientes en términos de accesos a disco[5].

2. Arboles Balanceados

Los árboles B son árboles multicamino con una construcción especial que permite mantenerlos balanceados a bajo costo. Un árbol B de orden M posee las siguientes propiedades básicas [6]:

- Cada nodo del árbol puede contener como máximo M descendientes y M-1 elementos.
- La raíz no posee descendientes o tiene al menos dos.
- Un nodo con k descendientes contiene k-1 elementos.
- Los nodos terminales tiene como mínimo [M/2] – 1 elementos y como máximo M -1 elementos.
- Los nodos que no son terminales ni raíz tienen como mínimo [M / 2] elementos.
- Todos los nodos terminales se encuentran al mismo nivel.

Un árbol balanceado tiene como principal característica que todos los nodos terminales se encuentran a la misma distancia del nodo raíz. De esta forma, se puede asegurar que la performance de búsqueda dentro de esta estructura siempre se encuentra acotada por el orden de búsqueda logarítmico con base equivalente al orden del árbol $O(\log_M n)$. Si se tratara de un árbol de balanceado de orden 256, con un millón de elementos que lo conformen, en el peor de los casos la altura del árbol (y por ende la cantidad de accesos máxima necesaria para recuperar la información) está acotado a 4.

La forma de construcción de un árbol balanceado difiere en gran medida de un árbol binario. Este tipo de árboles no puede crecer hacia abajo, como ocurre con los binarios, debido a que esta construcción es la que genera el desbalanceo de la estructura. La construcción (inserción) parte desde los nodos hojas y se va extendiendo hacia la raíz. Nuevamente, este trabajo no tiene por objetivo definir en detalle el proceso de construcción de un árbol balanceado. Bibliografía como [1], [4] y [6] describen en detalle esta metodología y discute, además, el mecanismo de búsqueda y eliminación.

Como desprendimiento de los árboles B, aparecen tanto los árboles denominados B* y B+. Los primeros son árboles B de construcción especial donde cada nodo se completa hasta 2/3 de capacidad como mínimo, aumentando el porcentaje de ocupación de cada nodo. De esta forma, los árboles B* plantean una alternativa a los

árboles B generando una estructura más eficiente en cuanto a la utilización de espacio.

Los árboles B* son árboles que al ocupar cada nodo con más elementos crecen más lentamente. La ventaja de este tipo de estructuras radica en que la altura es menor y por consiguiente la performance de búsqueda de un elemento es mejor. Nuevamente, en el marco de la teoría y práctica de la asignatura Introducción a las Bases de Datos, se discute detalladamente la metodología y algoritmos necesarios para el proceso de búsqueda, inserción y eliminación sobre árboles de tipo B*.

Por último, en el marco metodológico de la materia, se presentan los árboles B+. Este tipo de estructuras presenta dos ventajas: la búsqueda eficiente de información (a través de un árbol balanceado) y el acceso secuencial a bajo costo (a través de un conjunto de nodos enlazados). De esta forma es posible asegurar el cumplimiento de los dos preceptos básicos que definen los índices, que son la búsqueda aleatoria eficiente de información y poder acceder a la misma en forma ordenada. Los árboles B+ son una clase particular de árboles B donde se cumplen las siguientes propiedades [6]:

- Cada nodo del árbol puede contener como máximo M descendientes y M-1 elementos.
- La raíz no posee descendientes o tiene al menos dos.
- Un nodo con k descendientes contiene k-1 elementos.
- Los nodos terminales tiene como mínimo [M/2] - 1 elementos y como máximo M -1 elementos.
- Los nodos que no son terminales ni raíz tienen como mínimo [M/2] descendientes.
- Todos los nodos terminales se encuentran al mismo nivel.
- Los nodos terminales representan un conjunto de datos y están vinculados constituyendo al mismo tiempo una estructura lineal (lista enlazada)

3. Marco teórico de la Propuesta

La propuesta de este trabajo consiste en generar una herramienta, que se denominó HEA, que sirva para asistir al alumno en el proceso de aprendizaje de árboles balanceados como alternativa para implantar estructuras de índices para las BD.

A partir de la actividad docente registrada de más de 10 años en un curso básico de Bases de Datos, se ha observado que si bien los alumnos comprenden el proceso de creación, eliminación y búsqueda sobre árboles balanceados, generalmente resultan escasos los ejemplos presentados a fin de mostrar todas las alternativas que ocurren en la práctica.

De esta forma, los alumnos resuelven los problemas de manera individual generando posteriormente consultas con los auxiliares de cátedra con el fin de corroborar la resolución de los ejercicios. Sin embargo, no siempre es posible evacuar todas las dudas debido a que hay situaciones puntuales que generan dudas sobre los alumnos, y estas se plantean fuera de los horarios de consulta.

El desarrollo de esta herramienta intenta complementar la actividad de enseñanza/aprendizaje de este tema. Así, el alumno puede analizar la resolución de un problema, generando el caso de uso y comprobando su resolución paso a paso.

HEA (Herramienta para la Enseñanza de Arboles) es una herramienta de software concebida para asistir al alumno en el proceso de aprendizaje, a partir del trabajo interactivo y con alta calidad de presentación visual, que permite construir, buscar o eliminar elementos de un árbol balanceado (en cualquiera de sus versiones: clásica, B* o B+) con las claves definidas por el alumno.

4. HEA: Herramienta para la Enseñanza de Arboles

HEA fue proyectado para ser una herramienta WEB. Desarrollada con software de uso libre y con concepción de software Open Source, busca ser un producto que pueda ser utilizado en cualquier contexto educativo de árboles B, como complemento para el proceso de enseñanza-aprendizaje.

Una vez formulada la propuesta, la primera etapa consistió en implantar todos los algoritmos necesarios para la administración de árboles balanceados. Los algoritmos definidos fueron los de creación e inserción de elementos, búsqueda de información y eliminación de información para árboles B, B* y B+ [4]. Para facilitar el desarrollo de la herramienta, en todos los casos, los elementos que componen el árbol son números enteros.

El proceso de diseño de la herramienta busca generar un producto que sea de fácil interpretación por el alumno y que lo asista en el proceso de generación, consulta o eliminación sobre una estructura de índices organizada mediante cualquier variante de un árbol balanceado.

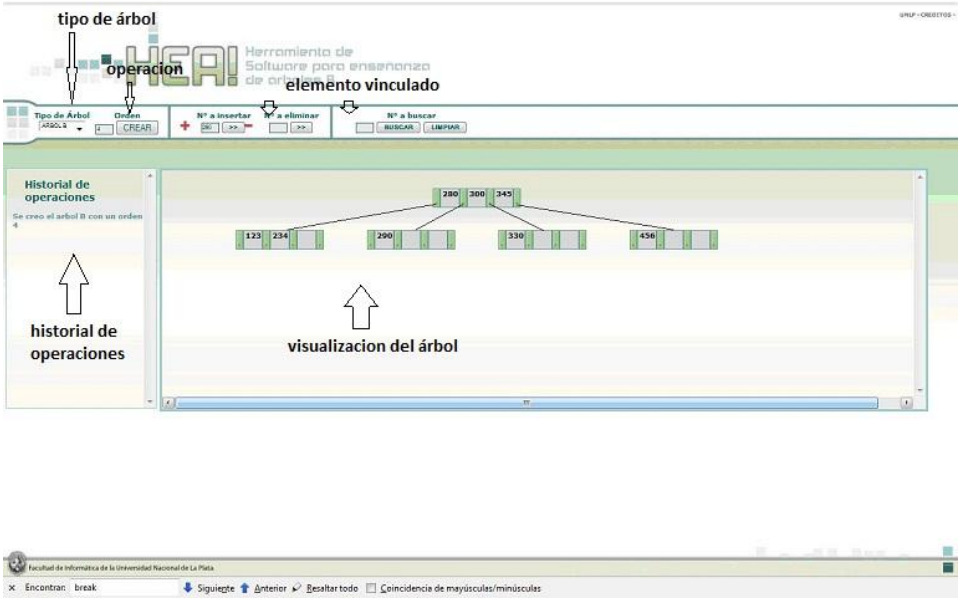

Figura 1

La figura 1 presenta un prototipo inicial de la herramienta. El primer paso en la utilización de HEA consiste en definir el tipo de árbol que se desea generar. Al definir el tipo de árbol se selecciona, automáticamente, los algoritmos de trabajo involucrados. El segundo paso consiste en seleccionar el orden del árbol. De esta manera, al crearse el árbol, el alumno podrá observar el nodo raíz con capacidad para almacenar (M-1) elementos, siendo M el orden seleccionado. Por otro lado se podrá ver que la raíz contiene (M) punteros vacíos, cuya cantidad también esta dada por el orden que selecciono el alumno.

Para los árboles B, el alumno tiene la posibilidad de indicar, ante una división de un nodo, donde quedarán mayor cantidad de elementos. Si por ejemplo el orden del árbol es 10, un nodo tiene capacidad para 9 elementos y 10 punteros. En caso de producirse saturación, con la llegada del décimo elemento, el nodo debe ser dividido. Al tener una cantidad par de elementos y promocionar la clave intermedia al nodo padre, quedan para repartir 9 elementos entre el nodo dividido y el nuevo, 4 irán a un nodo y 5 al restante. El alumno debe definir como parte de las condiciones del problema cual de los nodos tendrá 4 elementos y cual tendrá 5.

En caso de tratarse de árboles B*, de acuerdo a la bibliografía utilizada por la cátedra, este tipo de árboles tiene varias alternativas. En primer lugar, que debe indicar el alumno, es la política de trabajo que tendrá el árbol B*. Estas políticas definen con que nodo hermano adyacente [6] se debe realizar la redistribución o división en caso que se produzca saturación en un nodo. Las políticas pueden ser tres:

- Un lado: indica si el nodo hermano adyacente con el que se redistribuye o divide es hacia izquierda o derecha.
- Un lado ***u*** otro lado: indica cual nodo hermano adyacente utilizar primero para redistribuir. En caso de no poder hacerlo con ese hermano se intentará redistribuir con el otro hermano adyacente. Si no se pudiera redistribuir con ninguno, el primero será tomado para dividir.
- Un lado ***y*** otro lado: en general, esta política es similar a la anterior. En caso que ambos hermanos adyacentes estén completos y no se pueda redistribuir, la operación de división involucra al nodo saturado y a ambos hermanos, tomando así tres nodos llenos y generando cuatro nodos tres cuartos llenos.

El segundo parámetro que el alumno debe definir es cual nodo contendrá más elementos (o menos) de acuerdo al orden del árbol y luego de efectuar la división. Así, si el orden fuera 10, y se utilizara la política de un lado, la división se genera cuando dos nodos están completos (20 elementos en total) y ante la llegada de un nuevo elemento. Se cuentan entonces con 22 elementos (los 20 de los nodos completos, el elemento nuevo, y el elemento que se encontraba en el padre y que actuaba como separador). Al dividir, y generar tres nodos, serán necesarios dos separadores. Por este motivo quedan 20 elementos para repartir entre tres nodos, uno de ellos contendrá 6 y los dos restantes 7. El alumno debe decidir cual de los nodos tendrá, en este caso, menos elementos para que el árbol se genere de acuerdo a sus expectativas.

Por último, los árboles B+ se plantean bajo el formato de construcción de los árboles B. La diferencia, en este caso, es que el formato de los nodos terminales varía, pues los mismos son enlazados entre sí formando una estructura lineal de tipo

lista. El único parámetro que el alumno debe definir para la creación de un árbol B+ es el mismo que para un árbol B.

En la figura 1 se observa, además, la operación que se va a llevar a cabo, en este caso, la operación definida es crear un árbol B.

La parte central de la interfaz muestra como se va generando el árbol a partir de las claves ingresadas. Estas claves se ingresan manualmente por parte del alumno y la ejecución de una inserción se realiza gradualmente (paso a paso) para que pueda observar el proceso, identificando porque se llega al resultado final.

Por último, se mantiene una traza o historial de ejecución de usuario. Esta traza permite al alumno retornar a algún punto anterior del proceso para revisar o repasar lo que ocurría en ese momento.

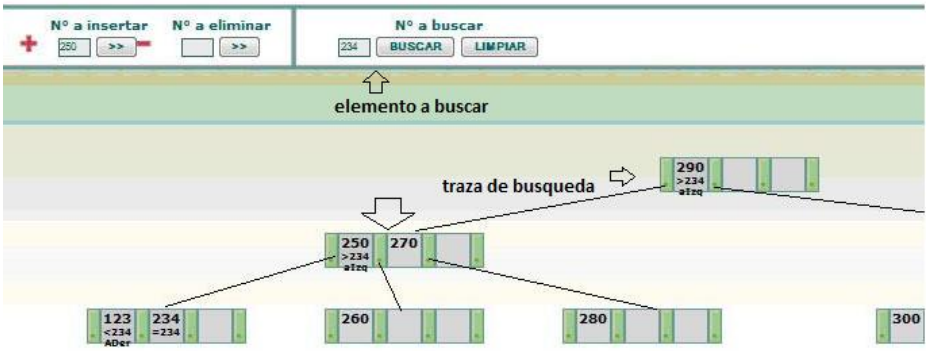

Figura 2

La figura 2 presenta el prototipo de la traza para el proceso de búsqueda de un dato en el árbol generado. El primer paso consiste en definir el elemento a buscar y luego, sobre el área de trabajo, se presenta la traza de búsqueda hasta detectar el elemento o hasta llegar a un nodo terminal sin haber encontrado el dato indicado.

La figura 3 presenta en forma gráfica el prototipo de división de un nodo en un árbol B. La inserción del elemento 200 produce saturación en el nodo donde debería residir. En la figura se presentan los elementos coloreados. En color verde, y de acuerdo a la forma de división propuesta por el alumno, figuran los elementos que quedarán en un nodo, en tanto que en celeste quedan los restantes. El elemento 220, que se muestra en tono gris, es promovido hacia el padre. Se puede observar, además, a la izquierda el historial de operaciones, que presenta toda la traza de lo actuado sobre el árbol generado.

En el análisis de la construcción de HEA, se puso especial atención a la facilidad de acceso a la herramienta, el uso de tecnologías con licencias libres y la necesidad de realizar animaciones con el fin de lograr un ambiente de aprendizaje claro y amigable al uso.

En función al primer punto a analizar (fácil acceso), se decidió que sea una herramienta WEB. Vale aclarar que decimos que HEA es de acceso WEB ya que se utiliza a través de un navegador, sin embargo, no es imprescindible un servidor WEB, ya que debido a la tecnología utilizada, la cual será explicada mas adelante, todo el

procesamiento es realizado del lado del cliente. De esta manera, si bien HEA estará alocado en un servidor, lo que permitirá al alumno acceder a través de la WEB, si el mismo no contara con acceso a Internet, podrá tener el sistema en su computadora y utilizarlo sin ningún inconveniente.

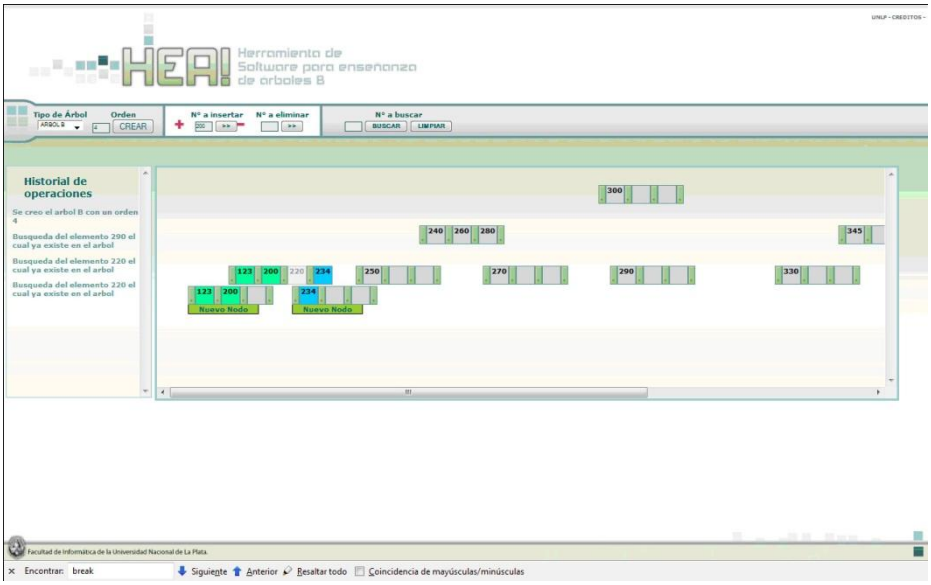

Figura 3

La figura 4 presenta otro caso gráfico de un nodo que ante una inserción debe ser dividido. Nuevamente se puede observar dinámicamente el proceso de división, con los colores ya definidos.

Teniendo en cuenta la escalabilidad de la herramienta se ha decidido seguir el MVC (Model View Controller), es decir, un desarrollo en capas, donde cada una de estas, tiene una tarea especifica que hacen al objetivo común pero logran cierta independencia lo que facilita el mantenimiento del sistema. Siguiendo este criterio todo lo que respecta a la lógica de la estructura del árbol es manejada por la capa de Modelo la cual interactúa con el Controlador quien es el encargado de encolar operaciones que serán utilizadas por la Vista en el momento de llevar a cabo la animación.

Para la programación, y respetando el segundo punto a analizar (tecnologías con licencias libres), se ha decidido utilizar los siguientes lenguajes: HTML (HyperText Markup Language) y JavaScript. Si bien, excede el alcance de este documento, resulta necesario hacer hincapié sobre los problemas de compatibilidad entre los diferentes navegadores con los que se encuentra un programador a la hora de construir una herramienta WEB. Para evitar este problema la programación de HEA esta pensada siguiendo el DOM (Document Object Model), que es un estándar cuyo responsable es el consorcio W3C (World Wide Web Consortium).

En la investigación sobre herramientas que permitieran la realización de animaciones, y sin dejar de lado el uso de tecnologías con licencias libres, se decidió

utilizar una biblioteca potente y flexible de javascript denominada Jquery. Esta librería permite lograr el efecto de transición de un nodo a lo largo de todas las operaciones que el alumno realiza sobre el árbol. A través de la utilización de Jquery se logra un efecto de animación al momento de crear un nuevo árbol, al producirse la división de un nodo, en la fusión y al momento de promocionar o borrar una clave del árbol.

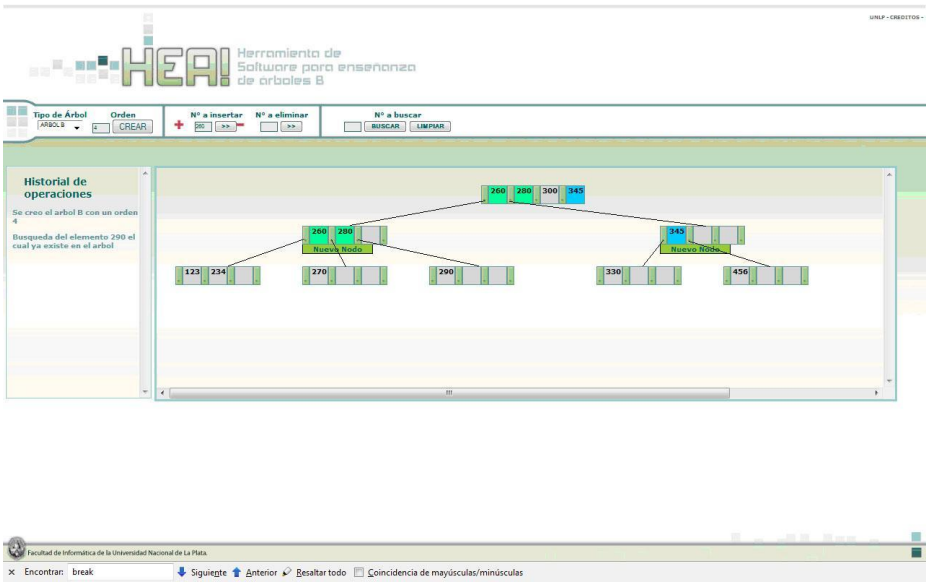

Figura 4

La parte estética de HEA, fue pensada teniendo en cuenta que se trata de una herramienta educativa. De esta forma lograr un ambiente amigable, de fácil lectura y de simple utilización fueron objetivos a cumplir. Para lograr una separación entre la lógica de programa y la parte estética se utilizaron hojas de estilo en cascada CSS (Cascading Style Sheets).

5 Conclusiones

La familia de los árboles balanceados son estructuras muy poderosas y flexibles para la administración de índices de una BD, con un desempeño a nivel performance óptimo. En particular, los árboles B+ permiten integrar la búsqueda eficiente de información, con el recorrido secuencial del archivo a bajo costo de mantenimiento

Uno de los objetivos previstos en la asignatura Introducción a las Bases de Datos es que el alumno comprenda los beneficios de este tipo de estructuras en la organización de los índices de una BD. Para ello, es importante que los algoritmos de inserción, búsqueda y eliminación de claves en un árbol sean analizados y comprendidos hasta el mínimo detalle.

La construcción de HEA busca proveer de una herramienta interactiva que permita al alumno comprobar de manera dinámica los ejercicios definidos en la práctica, a fin de agilizar el proceso de aprendizaje. Además, el alumno puede plantear sus propias variantes prácticas, analizando, por ejemplo, la diferencia en la construcción de un árbol B* a partir de seleccionar otra política de trabajo con los nodos hermanos adyacentes.

La herramienta actualmente está en una etapa de prueba. Los prototipos desarrollados hasta el momento se han planteado en reuniones de brainstorming dentro de la cátedra. La evolución actual del producto permite suponer que durante el año 2011 HEA estará disponible para ser utilizado en la asignatura, y por ende, probada por más de 600 alumnos.

6 Trabajos Futuros

Los trabajos futuros son variados. En una primera etapa, y luego de una prueba de campo donde se esperan depurar posibles problemas de interfaz, se plantea agregar al prototipo la opción de trabajar con árboles B+ de prefijos simples. Esta alternativa, la más utilizada en general por los DBMS de mercado, llevará a utilizar registros de longitud variable. Esto no representa en si un problema, sino un cambio en el enfoque de tratamiento de las estructuras que soportan HEA.

Además, se prevé la utilización de un orden diferente para los nodos terminales y no terminales en un árbol B+ de prefijos simples. Estos temas son desarrollados parcialmente en la materia, pero al contar con una herramienta que muestre la evolución de los nodos del árbol de manera dinámica, se cree que el proceso de aprendizaje de esta estructura será mucho más simple.

7 Referencias

1. Files & Databases: An Introduction. Peter Smith, Michael Barnes. Addison Wesley 1987.
2. Fundamento de Sistemas de Bases de Datos. Elmasri, Navate. Addison Wesley. 2002.
3. Fundamentos de Bases de Datos, 5ª edc. Sudarshan S.; Korth Henry; Silberschatz Abraham. McGraw-Hill. 2008
4. Estructuras de Archivos. Michael Folk, Bill Zoellick, Greg Ricciardi. Addision Wesley. 1992
5. Availabilities and costs of reliable Fat-Btrees. Miyazaki, J. Abe, Y. Yokota, H. Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium on Issue Page(s): 163 – 172. 2004
6. Introducción a las Bases de Datos. Fundamentos y Diseño. Rodolfo Bertone, Pablo Thomas. Pearson Latinoamerica. En proceso de edición. Año 2010.