INDEXACION DIRECTA DE INFORMACIÓN TEMPORAL PARA EL POSTGRES IDITPos

Juan Fernando Vélez M.*
Marta Lucía Aristizábal**
Robinson Coronado**
Wilmer Quiceno**
William Alexis Ortiz**
Mauricio Zapata Vanegas**

ABSTRACT

In this article presents the DIRECT INDEXING OF TEMPORAL INFORMATION POSTGRES (IDITPos) investigation project. The project development was based on the database system manager POSTGRES, the mechanism of storage and temporal information retrieving of the queries engine. Also was defined a basic cost model for storage and query of historical data. The POSTGRES uses the R tree to handle the historical information. An interesting work for doing would be to add to engine the Direct Indexing method to improve the access to those data. To realize it, was necessary to compare mathematical the cost of historic queries using both, R Tree and direct indexing. The project is framed into the "Temporal Data Base". These system are obtained form the traditional Relational database adding the time property to the relations. A system which integrate this Temporal Dimension allow to realize queries about the data base past.

RESUMEN

Teniendo en cuenta que nuestro país debe apropiarse de las nuevas tecnologías informáticas, nuestra universidad viene desarrollando un frente de investigación en el campo de las Bases de Datos. En particular, el Departamento de Ingeniería de Sistemas de la Universidad de Antioquia ha adoptado como estrategia emprender proyectos que permitan conocer y mejorar las funciones de los actuales sistemas de bases de datos de dominio público.

El desarrollo del presente proyecto se basó en el estudio del sistema de gestión de bases de datos denominado POSTGRES[1], concebido en la Universidad de Berkeley para proyectos de investigación.

* Ingeniero de Sistemas, Universidad de Antioquia
Maestría en Informática, Universidad de Pierre et Marie Curie – París VI
DEA en Bases de Datos, Universidad París 1, Pantheon – Sorbonne
En la actualidad, estudios de Tesis Doctoral en el área de las Bases de Datos Temporales, Universidad de París 1, Pantheon – Sorbonne.

* Estudiantes Ingeniería de Sistemas.
Grupo de Investigación en Bases de Datos, Facultad de Ingeniería, Universidad de Antioquia.

[1] La descripción técnica del POSTGRES aparece en otro artículo de esta revista.

Se estudiaron los mecanismos de almacenamiento y recuperación de información temporal del motor de consultas, se definió también un modelo básico de costos para el almacenamiento y la consulta de datos históricos.

El POSTGRES utiliza el árbol R para manipular la información histórica. Un trabajo interesante sería el adicionarle al motor el método de indexación directa para mejorar el acceso a tales datos. Para llevarlo a cabo, fue necesario comparar matemáticamente el costo de las consultas históricas utilizando tanto el árbol R como la indexación directa.

Este proyecto se enmarca dentro de las "Bases de datos temporales". Estos sistemas se obtienen de las tradicionales bases de datos relacionales agregándole a las relaciones el atributo de tiempo. Un sistema que integra esta dimensión temporal permite entonces realizar consultas sobre el pasado de la base de datos.

1. CONSIDERACIONES GENERALES

1.1. JUSTIFICACIÓN

La elección del ambiente POSTGRES se basó en el hecho de que este software es de dominio público y además fue diseñado con el objetivo de servir como base para el desarrollo de investigaciones en el campo de los sistemas de gestión de bases de datos.

Al trabajar con este tipo de software se pueden realizar estudios basados en simulaciones sobre los diferentes componentes del motor, es decir, sobre el optimizador, sobre los métodos de acceso, los mecanismos de almacenamiento, entre otros. Dependiendo del resultado de estas simulaciones se pueden generar nuevas estrategias para intervenir el código original del programa.

1.2. OBJETIVOS

- Definir un modelo básico de costos para consultas históricas utilizando dos métodos de acceso: el árbol R y la indexación directa mediante transformaciones lineales.
- Definir una estrategia para implementar nuevos métodos de acceso a datos históricos en el POSTGRES.

2. ASPECTOS TEÓRICOS

2.1 BASES DE DATOS TEMPORALES

2.1.1 Representación del Tiempo

La dimensión temporal se puede representar como un conjunto de puntos de tiempo (instantes) consecutivos y equidistantes de la siguiente forma:

T = {0, 1, 2, ..., now},

donde :

0 es el punto de partida de la aplicación real

now es el tiempo corriente.

La granularidad entre dos instantes de tiempo puede ser: años, meses, semanas, días, horas, minutos, segundos, etc. Además se pueden realizar las siguientes definiciones:

Intervalo de tiempo	Período de Vida. Se denota por [ts, te] como subconjunto de T, donde ts es el tiempo inicial y te es el tiempo final del intervalo.
Elemento temporal	Es la unión finita de intervalos de tiempo denotado por {I1, I2, I3, ...}, donde Ii pertence al intervalo universal T=[0,now]. Si se tienen dos elementos temporales pertenecientes a T se pueden definir las operaciones entre conjuntos: unión, intersección, diferencia y complemento de T, y también los predicados de comparación utilizando =, ≠, ⊇ y ⊆.

2.1.2 Clasificación de las Bases de Datos

Los sistemas de gestión de bases de datos, SGBD, se pueden clasificar tomando en cuenta su capacidad

para soportar las dimensiones temporales, las cuales son:

El tiempo de validez	Intervalo de tiempo durante el cual la información almacenada en la base de datos es válida en el mundo real. Esta dimensión permite informarse sobre el pasado o anticiparse al futuro.
El tiempo de transacción	Da cuenta del momento en que la información es registrada en la base de datos. Esta dimensión traduce la historia de las modificaciones sobre la base de datos.

Por tanto, se presentan cuatro tipos de bases de datos:

- Los SGBD clásicos que no soportan el tiempo de validez ni el tiempo de transacción son llamados bases de datos estáticas (*snapshop database*). Estos sistemas solo dan una visión instantánea de los datos aplicados de la aplicación real.
- Cuando el SGBD soporta el tiempo de validez se habla de una base de datos histórica (*historical databases*).
- Cuando se dispone de un soporte para el tiempo de transacción se habla de bases de datos retrospectivas (*rollback databases*).
- Si el SGBD soporta simultáneamente las dos dimensiones temporales, se habla entonces de bases de datos bitemporales (*bitemporal databases*). Permiten realizar operaciones de actualización retroactivas (corrección de errores) en los datos históricos.

2.1.3 Representación Espacial de una Relación Temporal

Una relación temporal puede ser representada en un espacio de dos dimensiones donde el tiempo es una de ellas. En la Figura 1, el eje horizontal representa la dimensión temporal [0,now] que contiene los puntos de tiempo discretos y el eje vertical es una dimensión que permite representar los valores para identificar las tuplas.

Tupla	Pid	PE	TV
T_1	P_1	NY	[0,3]
T_2	P_1	LA	[4,Now]
T_4	P_2	SFO	[0,5]
T_7	P_3	LA	[0,7]
T_8	P_3	SFO	[8,9]
T_{10}	P_4	NY	[2,3]
T_{11}	P_4	LA	[8,now]
T_{12}	P_5	LA	[10,now]
T_{13}	P_6	NY	[12,now]
T_{14}	P_7	NY	[11,now]

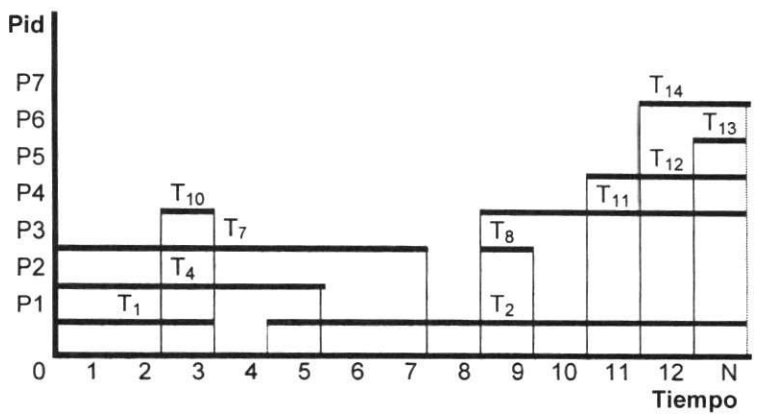


Figura 1

2.1.4 Modelo de Almacenamiento Temporal

En los SGBD temporales, es difícil mantener almacenados grandes volúmenes de información histórica en una sola zona de memoria – disco. Por lo tanto, la respuesta a consultas que implican la recuperación de datos históricos es ineficaz.

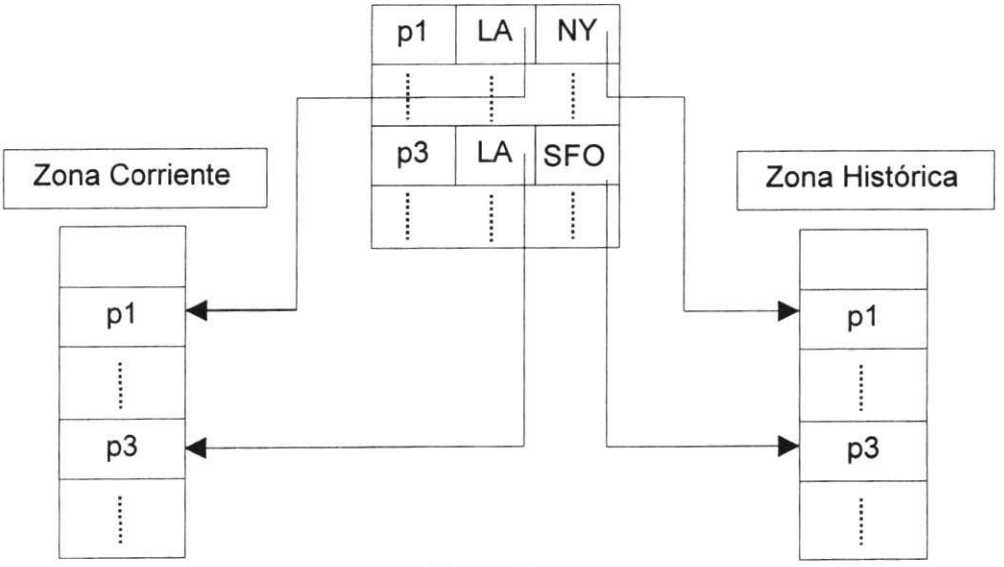


Figura 2

Para mejorar el acceso a la información histórica existen los métodos de almacenamiento temporal compartido (encadenamiento hacia atrás, acceso por lista o indexación), esta técnica consiste en almacenar los datos corrientes en una zona corriente y los históricos, en una zona histórica.

La zona corriente contiene las versiones corrientes que satisfacen todas las consultas no temporales y la zona histórica guarda las versiones históricas que permiten responder a las consultas temporales.

En la Figura 2 se observa que una entrada del índice puede contener información temporal que permite evaluar los predicados temporales sin hacer acceso a las tuplas de datos. La indexación es muy útil para manipular las tuplas que serán suprimidas. Como las versiones históricas tienen un camino de acceso independiente y no necesitan recorrer la zona corriente, todas las versiones pueden ser almacenadas en la zona histórica. Las futuras versiones podrían ser almacenadas, sea en la zona corriente, sea en la zona histórica y serían referenciadas por un índice.

2.2 EL ARBÓL R

Las Bases de Datos relacionales no están adaptadas para manejar objetos de tipo geométrico. Para hacer frente a tales necesidades se deben definir nuevas estructuras de almacenamiento llamadas *índices espaciales*.

Estas técnicas deben ser tenidas en cuenta para concebir nuevos optimizadores de consultas espaciales, que actualmente son un campo de intensa investigación. Una característica importante de estos índices es que pueden ser implementados para almacenar información temporal. Las consultas espaciales más utilizadas son:

- *El apuntado*: Se selecciona un objeto cuya geometría contiene el punto p.
- *El ventanazo*: Se seleccionan los objetos tales que su geometría corta o está contenida en una ventana.
- *El join espacial*: Dadas dos colecciones de objetos, se seleccionan las parejas tales que su geometría se intercepta.

2.2.1 Principio de la Indexación Espacial

Para simplificar las operaciones sobre los índices espaciales, la construcción del índice no se hace sobre el objeto mismo sino mas bien sobre su rectángulo mínimo englobante (REM). Se tiene entonces que las entradas (*REM, oid*) en los nodos hoja del árbol son ordenadas de manera que permiten seleccionar

los REM que contienen los objetos involucrados en las consultas. Por tanto, toda operación que utiliza el índice es ejecutada en dos partes:

1) Se recorre el índice para seleccionar los REM que conciernen la consulta (apuntado, ventanazo, join, etc.).
2) Se aplican las operaciones sobre los objetos (intersección y adyacencia de polígonos).

2.2.2 Definición del Árbol R

Es un método de indexación para objetos de tipo polígono. Esta técnica es la más utilizada en las bases de datos espaciales. El árbol R es una estructura equilibrada inspirada de los árboles B; al igual que estos árboles, el árbol R puede ser utilizado para almacenar objetos evolutivos y de distribución cualquiera. Es decir, el árbol evoluciona con operaciones de inserción y supresión de objetos. Posee las siguientes propiedades:

- Todo nodo del Arbol R está asociado con una zona rectangular de un espacio. Por tanto la raíz está asociada con todo el espacio. El rectángulo asociado a un nodo interno es el REM de los rectángulos asociados a cada uno de su hijos. Y el rectángulo asociado a cada entrada de un nodo hoja contiene un objeto (Polígono).
- Cada nodo del árbol, exceptuando la raíz contiene un número de entradas n tal que $m<n<M$. El límite inferior (o grado) m evita la degradación del árbol y asegura un almacenamiento eficaz de los objetos. El límite superior M se escoge tal que sea igual al tamaño de una página disco.
- Si el nodo raíz no es una hoja, tiene al menos dos entradas.
- El Árbol R es una estructura equilibrada, es decir todos los nodos hojas están al mismo nivel.
- Un nodo interno referencia los nodos del nivel siguiente en el árbol. Una entrada es una pareja (REM, prt) donde REM es el rectángulo del nodo hijo y prt su dirección.
- Un nodo hoja referencia el conjunto de objetos espaciales contenidos en su rectángulo. Una hoja contiene entradas (REM, oid) donde REM es el rectángulo que engloba el objeto espacial y oid es su dirección física en disco.

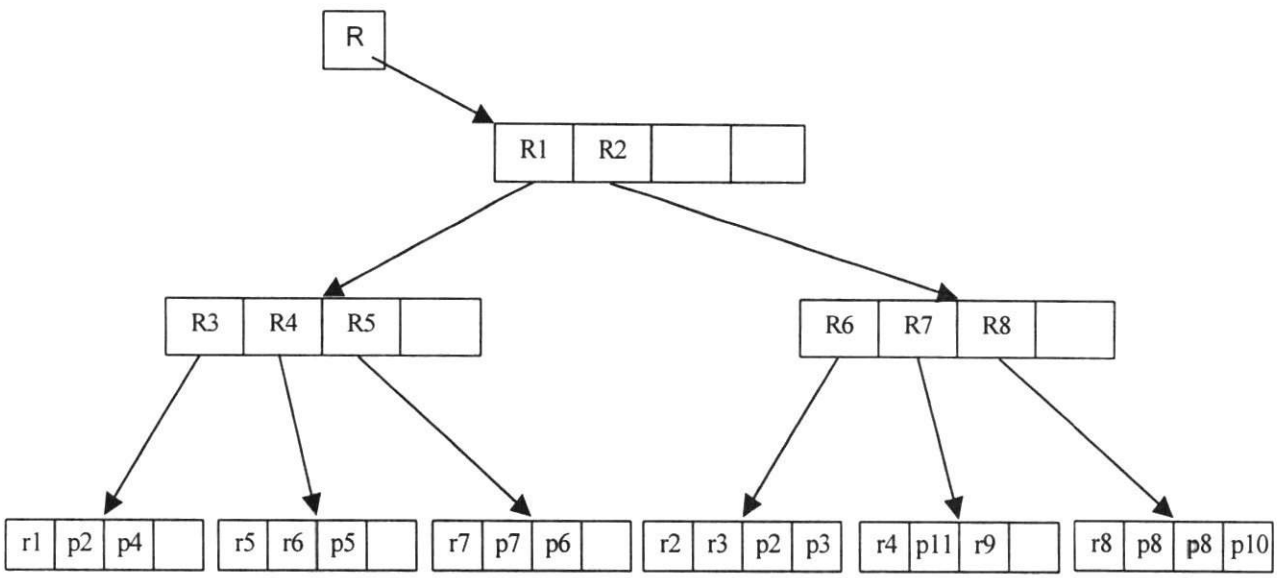


Figura 3.

En la Figura 3 se muestra la estructura de un árbol R de grado 2 construido sobre un conjunto de objetos geométricos. Puede verse que un objeto espacial es representado en una sola hoja del árbol y que los elementos que están en los nodos intermedios son rectángulos englobantes.

2.2.3 El Árbol R como Índice Temporal

El árbol R puede ser utilizado para almacenar relaciones temporales de la siguiente manera:

- Las entradas de un nodo no hoja son parejas (*REM, pt*) donde *REM* es el rectángulo englobante del nodo hijo y *pt* es el apuntador a tal nodo. Un REM esta definido por los valores máximos y mínimos de las dimensiones horizontal y vertical respectivamente.
- Un nodo hoja contiene parejas (*REM, id-tupla*) donde *REM* es el rectángulo del registro e *id-tupla* su dirección.

A continuación se presentan algunos inconvenientes al utilizar los árboles R para indexar relaciones temporales:

- Los árboles R no son apropiados para almacenar información temporal. Esta técnica funciona mejor para colecciones de objetos cuya naturaleza requiere un almacenamiento multidimensional, como es el caso de los objetos geométricos manipulados por los sistemas de información geográfica (SIG).
- Las relaciones temporales tienen atributos cuyos valores cambian independientemente del atributo de tiempo de validez. Por tanto, estas relaciones deben ser almacenadas con un método de acceso unidimensional.
- Cuando un árbol R es utilizado para almacenar una relación temporal, los algoritmos cortan excesivamente los datos (intervalos) almacenándolos en páginas diferentes. Por esta razón el rendimiento de este método se degrada.

2.3 INDEXACIÓN TEMPORAL DIRECTA CON ÁRBOLES B+

Para poner en marcha este tipo de indexación es necesario representar una relación temporal en un espacio multidimensional. En particular, por cada tupla se puede realizar una translación asociando el intervalo de tiempo de validez con un punto (x,y) en un espacio bidimensional.

Consideremos la siguiente relación temporal que hace referencia a las visitas de personas a Estados Unidos:

Tupla	Pid	PE	TV
T_1	P_1	NY	[0,3]
T_2	P_1	LA	[4,now]
T_4	P_2	SFO	[0,5]
T_7	P_3	LA	[0,7]
T_8	P_3	SFO	[8,9]
T_{10}	P_4	NY	[2,3]
T_{11}	P_4	LA	[8,now]
T_{12}	P_5	LA	[10,now]
T_{13}	P_6	NY	[12,now]
T_{14}	P_7	NY	[11,now]

A cada tupla μ perteneciente a la relación temporal se asocia un intervalo de tiempo de validez denotado $V(\mu)$. La duración de la visita de una persona se representa como un intervalo de tiempo en la dimensión temporal [0,now].

2.3.1 La Selección Temporal como una Búsqueda Espacial

Sea I el conjunto de todos los intervalos [a,b] en la dimensión temporal [0,now] y P el conjunto de puntos discretos de tiempo en [0,now].

La *Transformación Intervalo-Espacial*, denotada T, es una función de I hacia P_2 tal que T([a,b]) = (a, b-a). La función T puede ser aplicada a cualquier tipo de intervalo (tiempo de validez o tiempo de transacción). Las tuplas de la relación pueden ser representadas como puntos discretos de tiempo en un espacio bidimensional. Para ello, se define una función M sobre esta relación tal que para cada tupla μ: $M(\mu) = T(V(\mu))$. Por ejemplo, M(t10) = (2,1) puesto que el intervalo de tiempo de la tupla t10 es [2,3]. La figura siguiente muestra el resultado de aplicar la función M a la relación:

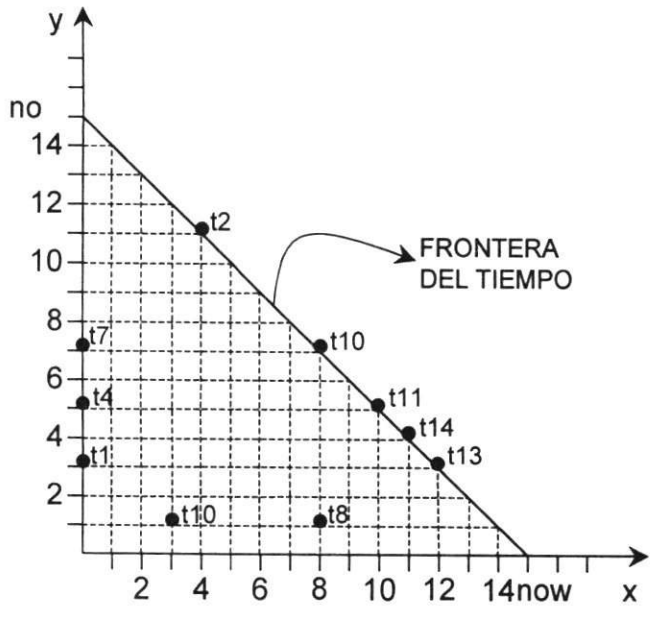


Figura 4.

Con esta representación gráfica pueden obtenerse tres parámetros asociados al tiempo de validez de cada tupla:

1) Una tupla con un tiempo de validez inicial $v_s = a$ caerá sobre la línea $x = a$
2) Una tupla con un tiempo de validez final $v_e = b$ caerá sobre la línea $x+y = b$
3) Una tupla con una duración total de tiempo de validez c caerá sobre la línea $y = c$

Teniendo en cuenta estas observaciones no es difícil entender como una consulta temporal sobre una relación puede ser transformada en una operación de búsqueda espacial en un espacio bidimensional.

Considere las siguientes consultas:

- Listar todas las personas que entraron al país antes del día t_a (o el mismo t_a).
- Listar todas las personas que salieron del país después del día t_b (o el mismo t_b).
- Listar todas las personas que permanecieron en el país durante t_c (o menos de t_c) días.

Las repuestas a cada una de estas consultas pueden ser encontradas recuperando todos los puntos que caen en las regiones mostradas en las figuras 5a), 5b) y 5c) respectivamente.

En este proyecto se supuso que la dimensión temporal es discreta. Una selección como "listar todas las personas que entraron al país antes del día t_a" puede ser fácilmente transformada en "listar todas las personas que entraron al país antes del día $(t_a - 1)$(o el mismo $t_a - 1$)".

Además, las conjunciones de criterios de selección corresponden naturalmente a intersecciones entre regiones, es decir, a criterios de selección individuales.

Por ejemplo, la consulta "listar todas las personas que entraron al país antes del día t_a (o el mismo t_a) y que partieron después del día t_b (o el mismo t_b)" es mostrada en la figura 5d) como la intersección de regiones de las figuras 5a) y 5b).

De manera similar, las disyunciones de criterios de selección pueden ser modeladas con la unión de las regiones respectivas. Por ejemplo la consulta "Listar todas las personas que entraron el día t_a (o el mismo t_a) o que permanecieron t_c (o menos de t_c) días en el país" puede ser satisfecha recuperando todos los puntos en la región mostrada en la figura e).

Finalmente, se considera una consulta que concierne una selección de puntos de tiempo. Por ejemplo, "encontrar todas las personas que estuvieron en el país el día t_f" corresponde al espacio de búsqueda en la figura 5f).

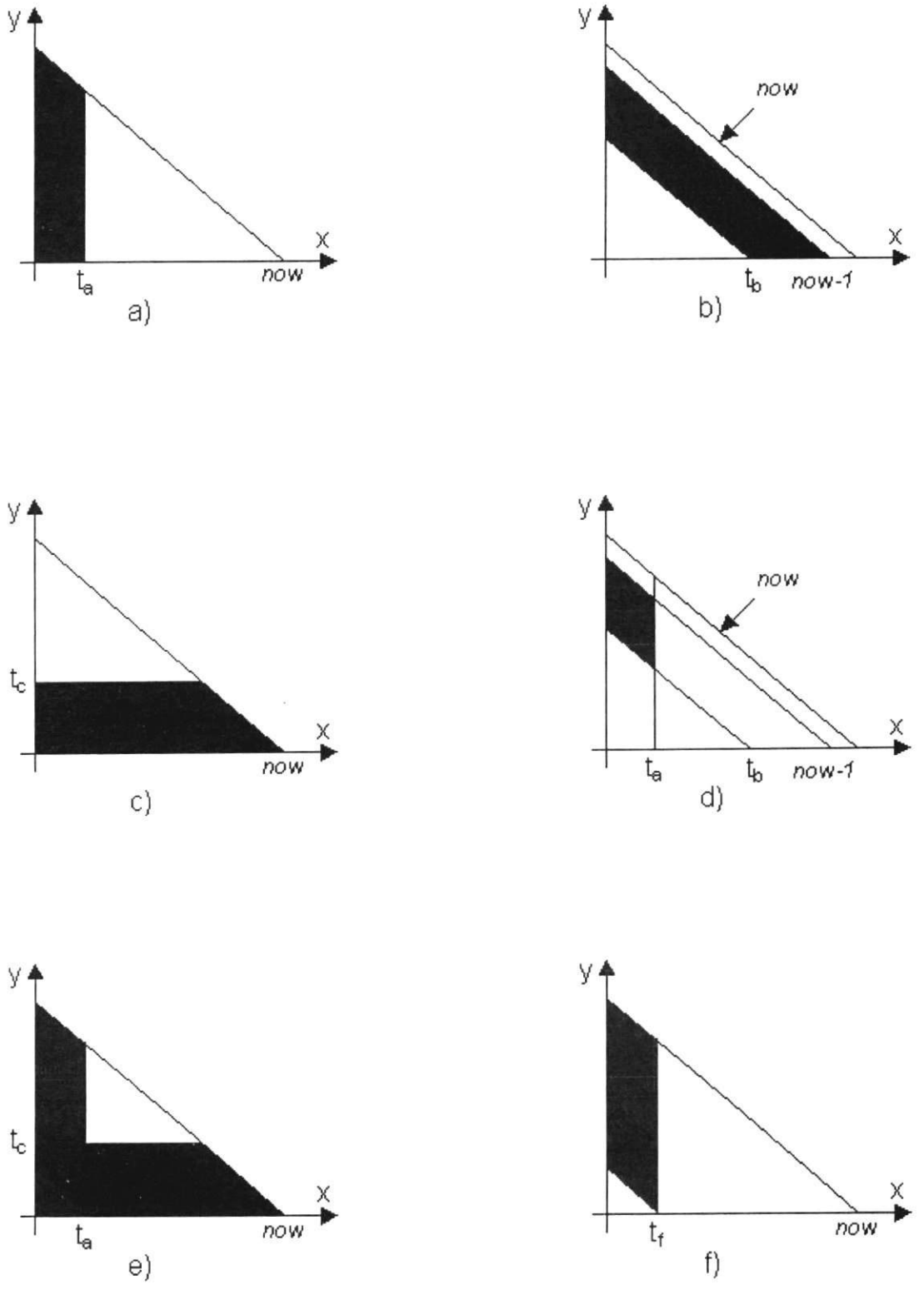

y
x
t_a
now
a)
y
now
x
t_b
now-1
b)
y
t_c
x
now
c)
y
now
x
t_a
t_b
now-1
d)
y
t_c
x
t_a
now
e)
y
x
t_f
now
f)

Figura 5.

2.3.2 Indexación de la Representación Espacial

El método consiste en explotar las características del espacio de búsqueda temporal. Dependiendo de los tipos de consultas temporales, con las mejores frecuencias de ejecución, es posible transformar puntos de un espacio bidimensional a un espacio unidimensional (lineal) utilizando diversos ordenes. Así, con estos puntos ordenados, se pueden utilizar los árboles B+ para indexarlos.

En la Figura 6 se grafican los órdenes lineales sobre los puntos en un espacio bidimensional P^2. Estos órdenes son el diagonal, el vertical y el horizontal respectivamente.

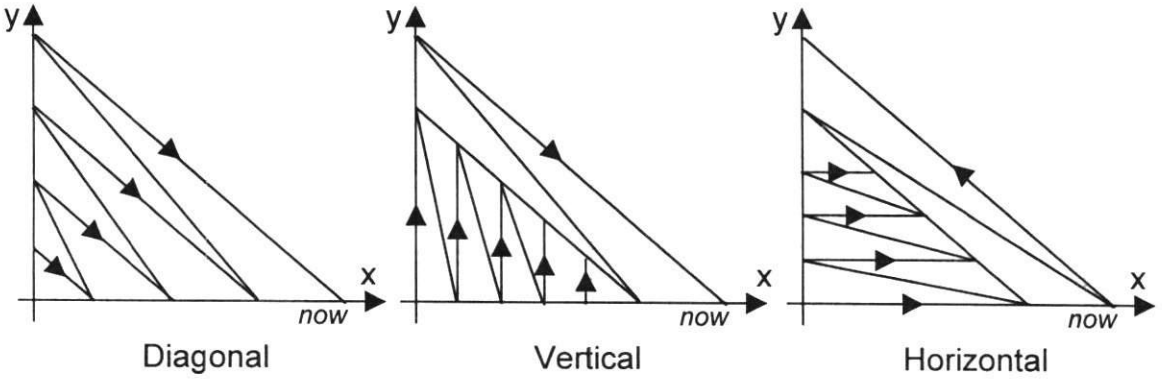


Figura 6.

Por tanto, las operaciones de búsqueda espacial sobre un espacio bidimensional pueden ser traducidas a operaciones de búsqueda sobre un espacio lineal definido por uno de estos órdenes.

En la Figura 7 se ilustra el árbol B^+ utilizado para indexar la representación espacial de la relación temporal que tomamos como ejemplo. En este caso se adoptó el orden diagonal.

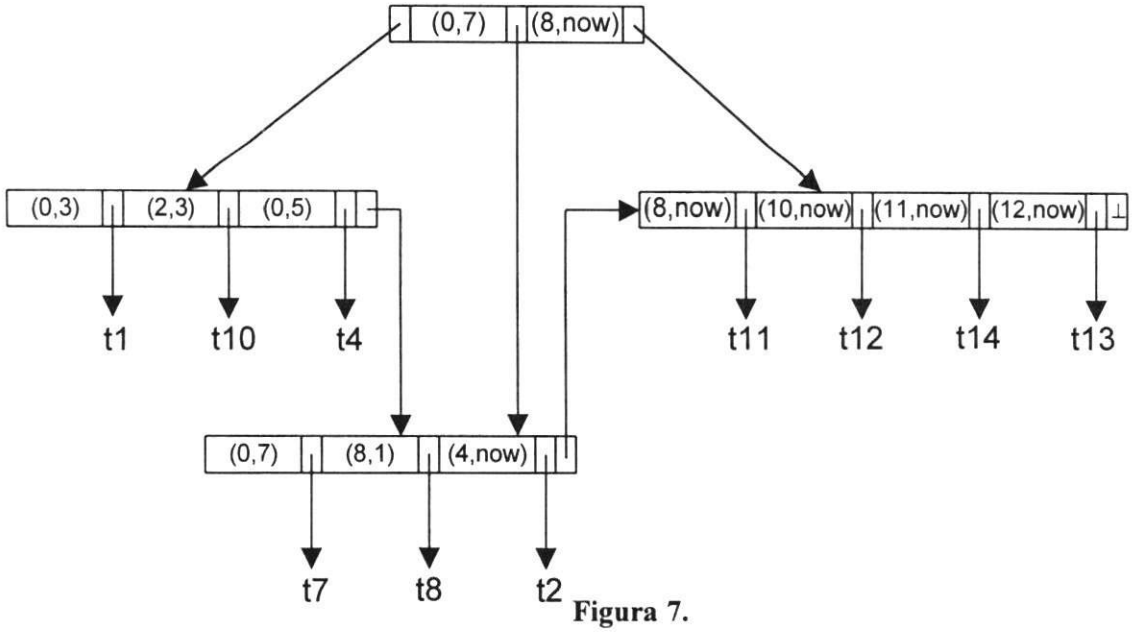


Figura 7.

3. RESULTADOS

A los dos métodos de acceso expuestos para relaciones temporales (la indexación directa con el árbol B^+ y el árbol R); se les realizó un análisis de rendimiento. Estos análisis se realizaron con base en simulaciones de las fórmulas de costo de ambos métodos. Estas simulaciones se hicieron con la herramienta SAS-IML.

El análisis consideró primero los parámetros del sistema con el fin de estimar el grado de los nodos de los árboles B^+ y R y luego se calculó:

- El costo de almacenamiento de relaciones, medido por el número de páginas disco utilizadas para guardar el archivo.
- El costo de las consultas sobre un intervalo de tiempo [a,b].

3.1 COSTO DE ALMACENAMIENTO DE UNA RELACIÓN TEMPORAL

La simulación que muestra la figura se realizó con base en el número de páginas a las que se hizo acceso por los índices (árbol R e indexación directa con árboles B^+) que se obtuvieron al variar el número de registros de una relación N.

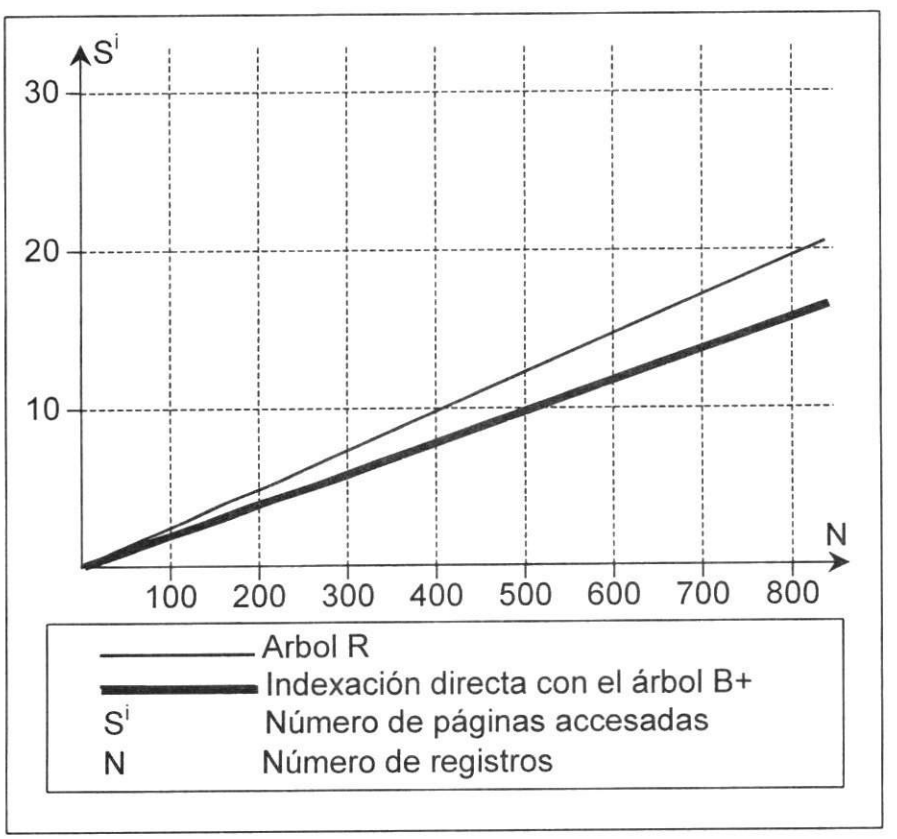


Figura 8.

3.2 COSTO DE UNA CONSULTA SOBRE UN INTERVALO

Este estudio del rendimiento de consultas consistió en estimar el número de accesos a páginas disco necesarias para entregar la información solicitada. Se realizó el análisis de una consulta sobre un intervalo I = [a, b] y se cálculo el costo con los dos métodos.

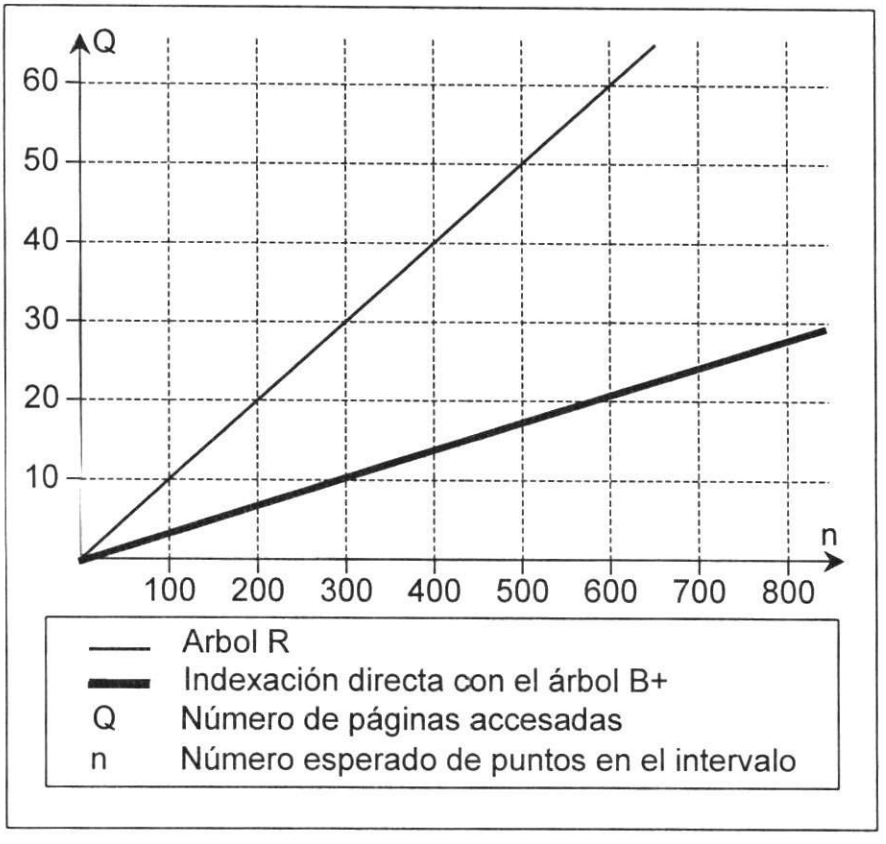


Figura 9.

4. PROTOTIPOS

4.1 IMPLEMENTACIÓN DEL ARBOL R PARA MANIPULAR INFORMACIÓN TEMPORAL

4.1.1 Introducción

El desarrollo de este prototipo, Se llevó a cabo en el marco del proyecto CODI[1] «Indexación directa de información temporal para el POSTGRES». La. implementación se realizó con el lenguaje C++.

Esta aplicación se basa en el almacenamiento del ARBOL R, propuesta por GUTMAN en 1984 [Gut84] con el objetivo de realizar simulaciones con grandes volúmenes de información temporal (tablas históricas).

Nuestro prototipo, desarrollado en los laboratorios de Ingeniería de Sistemas, mostró el manejo de versiones de objetos (tuplas de una relación histórica) geométricos, con su respectivo ordenamiento e inserción de los datos realizados en memoria.

[1] Comité para el Desarrollo de la Investigación, Universidad de Antioquia.

Se analizó el funcionamiento del árbol R en el POSTGRES, describiendo de una forma general los algoritmos que utiliza este gestor de base de datos para la eficiencia de su gestión.

Además, se realizó una descripción técnica de los módulos realizados en el prototipo, como las clases, instancias, acciones de la mismas y en general las estructuras presentes en este prototipo.

Se señalaron en forma detallada los aspectos para mejorar y tener en cuenta en una segunda fase del proyecto, en particular lo que concierne a la implementación de consultas, interfaces etc.

4.1.2 Descripción Específica de los Módulos del Prototipo

El prototipo genera una estructura de almacenamiento (árbol R) sobre la cual se van insertando datos. Estos datos son segmentos de línea que representan intervalos de tiempo de algún registro en particular, matriculado en alguna base de datos.

Para poder realizar todo lo anterior, el programa se apoyó en los siguientes módulos:

Clase Rtree: Es la estructura base del almacenamiento para los datos temporales generados.

La clase Rtree posee funciones miembros tales como:

BuscarDondeInsertar: Función que compara las coordenadas nuevas a insertar. Calcula la distancia mínima de algunos de los puntos constituyentes del segmento y lo envuelve donde esa distancia sea menor. Si el segmento cae dentro de algún rectángulo (por comparación de sus coordenadas) lo ubica adentro con base en la distancia mínima de éste.

Si no está adentro, calcula la distancia mínima del objeto con respecto a todos los rectángulos.

DistanciaPorFuera recibe las coordenadas nuevas a insertar, la dirección del nodo donde posiblemente se puede insertar el segmento y calcula la distancia.

MinFrontera: Calcula el mínimo rectángulo que puede envolver a otros REM(rectángulo mínimo envolvente) hallando la mínima fila y la mínima columna de los puntos iniciales de los REM y los mayores valores de los puntos finales de todos los REM.

Split: Realiza la partición de un nodo lleno, en el cual vamos a insertar un nuevo elemento.

En vectores guarda las coordenadas de los puntos inferiores y superiores de rectángulos del nodo, los apuntadores de sus hijos, y las coordenadas del nuevo objeto.

Se pasa luego a calcular la diferencia entre la menor posición en X inicial y la mayor posición en X final y calcula la distancia, lo mismo se hace con las Y *(función varianza)*. En donde sea la menor distancia se hace la división. Se ordenan las coordenadas de los objetos y se calcula la media *(función mediana)*; la mitad de los objetos quedan en el nodo viejo y la otra mitad se asignan al nodo recién creado. Se hace un nuevo cálculo de los rectángulos desde el nodo dividido hasta la raíz.

Insertar: Ubica el nuevo nodo en el REM correspondiente. Se apoya en las funciones de MinimaFrontera y BuscarDondeInsertar.

MayorVarianza: Determina sobre que eje se halla la mayor dispersión de los datos sobre la pantalla.

Liberar: Esta función libera uno a uno los nodos del árbol, al cual no se le van a realizar más inserciones.

4.1.3 Utilización del Prototipo del Árbol R

Inicialmente, el usuario encontrará un menú desplegable con las siguientes características:

Para iniciar una entrada de registros se oprime el botón CREAR, para salirse totalmente del programa se oprime el botón SALIR

Los cuadros horizontales indican el intervalo de tiempo inicial y final en que una tupla fue ingresada y retirada de la base de datos, generando un segmento

de tiempo. Los cuadros verticales son los identificadores de tuplas, identifican uno y solamente un registro dentro de la base de datos. El cuadro superior derecho permite ingresar el valor identificador de la tupla, el año de inicio y el año final.

Si se desea ingresar otro elemento a la base de datos, se oprime la tecla ENTER, si por lo contrario se desea ingresar mas registros se oprime la tecla ESC.

A medida que se ingresan intervalos de tiempo a algún registro, el programa va generando REM (rectángulos mínimos envolventes) siguiendo una debida lógica de ordenamiento.

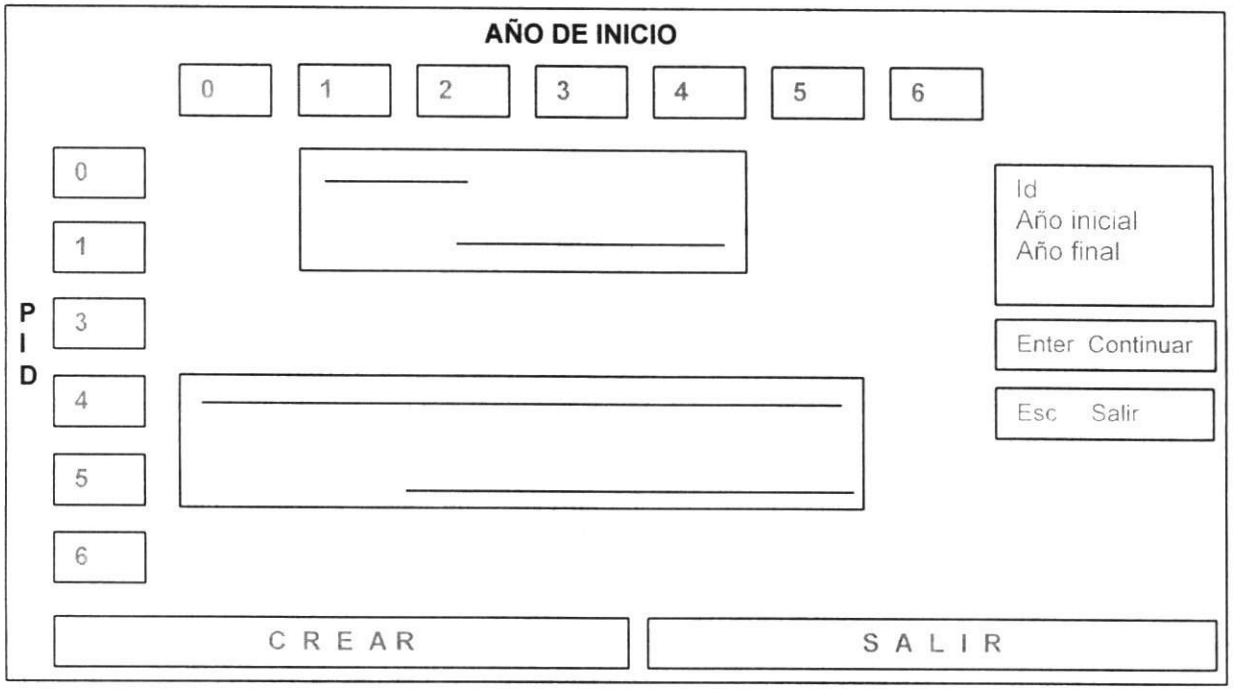


Figura 10. Interfaz del programa.

4.1.4 Perspectivas

Con este prototipo se pretende dejar los cimientos para intervenir en un futuro la herramienta POSTGRES implementando el índice temporal en la misma.

El prototipo es susceptible de mejoras, en aspectos tales como ordenamiento de datos, la partición de nodos, la interfaz con el usuario, etc.

Dentro de este prototipo no se han implementado consultas de ningún tipo, por consiguiente, se puede generar consultas como por ejemplo:

- Consultas puntuales.
- Consultas de rango como:
  - ✓ Consultas a la izquierda
  - ✓ Consultas a la derecha
  - ✓ Consultas sobre algún intervalo dado

Basándose en la filosofía, implementación y diseño de este prototipo, será fácil crear una estructura que administre la información de una base de datos temporal, mucho más eficiente, óptima y manejable, la cual será el árbol R^+.

4.2 PROTOTIPO DEL MÉTODO DE INDEXACIÓN DIRECTA UTILIZANDO EL ÁRBOL B+

4.2.1 Introducción

El diseño del prototipo fue realizado utilizando la herramienta C++ con los equipos proporcionados por el CODI en el proyecto «Indexación Directa de Información Temporal para el POSTGRES».

El objetivo del prototipo es afianzar los conocimientos básicos del manejo adecuado de la información temporal, además de facilitar la implementación de la indexación directa utilizando árboles B+ en el motor POSTGRES.

Para la creación de los árboles, se generan tuplas en forma aleatoria que son transformadas de acuerdo a tres tipos de ordenamientos (diagonal, vertical y horizontal) antes de ser insertadas en los diferentes árboles. Estos árboles son manipulados en memoria dinámica.

Se hace una descripción técnica de los módulos realizados en el prototipo, como las estructuras de datos, las funciones y los diferentes archivos utilizados.

Se espera, con la implementación del prototipo, enriquecer posteriormente el sistema POSTGRES con un mejor método de acceso para el manejo de la información temporal.

4.2.2 Especificaciones del Prototipo

Nombre del archivo: bmas.cpp. Contiene el desarrollo de los métodos utilizados para la creación y manipulación del árbol B+.

- **Métodos Incluidos**
  - ✓ **pila_vacia**

Estos métodos son utilizados para saber si la pila utilizada en la creación del árbol B+ se encuentra vacía.

- ✓ **init_pila**

Se utilizan para inicializar la pila que se ha creado.

- ✓ **cderecha_apunt**

Sirve para mover los apuntadores de una página dada hacia la derecha.

- ✓ **donde**

Esta función es utilizada para hallar la posición en la cual se debe insertar un dato en el árbol.

- ✓ **insertar**

Este procedimiento se encarga de insertar un dato en el árbol B+, además de su respectivo apuntador a disco.

- ✓ **ins_bmas**

Función principal que manipula la inserción en el árbol. Manipula la información desde la raíz y se encarga de llamar a los métodos necesarios para llevar a cabo la inserción. Recibe como parámetros la raíz del árbol, el apuntador a las hojas, el dato para insertar y su respectivo apuntador a disco, y envía como respuesta un valor que indica si el dato para insertar ya se encuentra en el árbol.

- ✓ **crear_pagina**

El objetivo de este procedimiento es la creación en memoria de una nueva página para el árbol B+. Recibe como parámetros el primer dato para insertar y su apuntador a disco, y envía al programa llamador la dirección de memoria de la nueva página.

- ✓ **inicializar**

Esta función se encarga de inicializar una página creada con el método crear_pagina dándole valores nulos a todos sus apuntadores, además de colocar el contador de datos en la página en cero.

- ✓ **buscar**

Este método es utilizado para conocer si un determinado dato que va a ser insertado en el árbol existe. Si el dato no existe entonces la función informa en cual posición debe insertarse el dato. Si el dato ya existe la posición devuelta será igual a un valor de -1.

✓ **romper**

Esta función es llamada en el momento en que una página no puede soportar más elementos. En ese instante la página debe ser dividida en dos páginas y se debe enviar el dato por subir a una página padre como consecuencia de la división.

✓ **ins_pila e ins1_pila**

Se utilizan para insertar en una pila la dirección de una página que se necesita más adelante. Las funciones de pila se utilizan para reemplazar los llamados recursivos en la manipulación de los árboles B^+ en este caso.

✓ **retira_pila y retira1_pila**

Estos métodos son utilizados para recuperar la información de una pila en la cual están almacenadas las direcciones de memoria de las páginas que se están manipulando actualmente.

✓ **retirar**

El procedimiento retirar se utiliza para "retirar" un dato de una página dada utilizando un desplazamiento hacia la izquierda y decrementando el contador de datos en una unidad.

✓ **cizquierda_apunt**

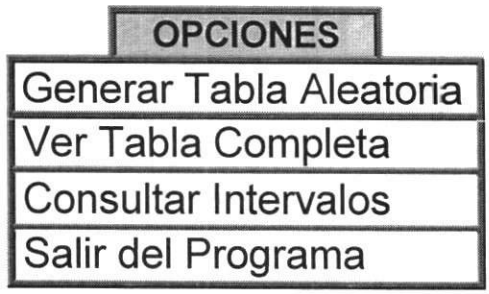
Sirve para mover los apuntadores de una página dada hacia la izquierda.

✓ **liberar**

Tiene por objeto liberar la memoria reservada para todas las páginas utilizadas en la construcción del árbol B^+. Es un método recursivo.

4.2.3 Utilización del Prototipo del Árbol B^+

Al ejecutar el programa aparece un menú de opciones que el usuario puede seleccionar utilizando las flechas de desplazamiento hacia arriba y hacia abajo. La pantalla muestra el siguiente contenido:

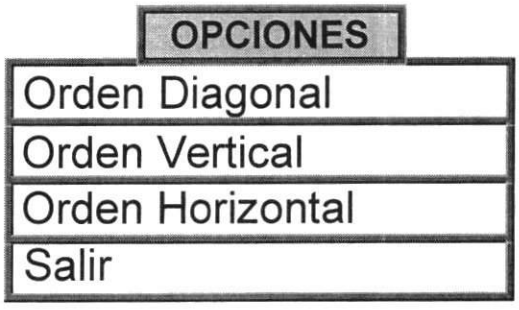


La opción Generar Tabla Aleatoria genera una tabla aleatoria en disco con el nombre de tabla.dat. Esta tabla contiene registros que están conformados por un identificador secuencial de los registros generados, además de los intervalos que siguen una distribución exponencial. Al ejecutar esta opción aparece un mensaje en pantalla indicando el número de registros generados aleatoriamente. Por omisión se están generando 1000 registros. Cada vez que el usuario seleccione esta opción se genera una nueva tabla que reemplaza la inmediatamente anterior.

La opción Ver Tabla Completa permite visualizar todo el contenido de la tabla generada. El proceso se realiza visualizando un conjunto dado de registros; si se quiere continuar la observación del siguiente grupo simplemente se oprime Enter. Si no se desean visualizar más registros se puede presionar la tecla Esc. La recuperación de la tabla no se realiza utilizando ningún tipo de índice; los registros se recuperan uno a uno en forma secuencial.

Para realizar consultas el usuario selecciona la opción Consultar Intervalos. Al seleccionar esta opción aparece en pantalla un mensaje indagando al usuario qué rango de intervalos desea para su consulta. El primer dato que se pide corresponde al inicio del intervalo de consulta, el segundo corresponde al final del intervalo. Luego de esta entrada de datos aparece un sub-menú.

Para salir del programa y devolver el control al sistema operativo se utiliza la opción Salir del Programa. El sub-menú mostrado en la opción Consultar es el siguiente:



El usuario puede seleccionar cualquier opción utilizando las teclas de desplazamiento hacia arriba y hacia abajo. Dependiendo de la opción seleccionada se hará una consulta utilizando un árbol indexado diagonal, vertical u horizontalmente. La opción Salir hace retornar el programa al menú anterior.

4.2.4 Perspectivas

Con base en la documentación y los conceptos adquiridos durante el proyecto, se espera llegar a entender la filosofía del manejo de la indexación dentro del POSTGRES, para luego implementar la indexación directa utilizando el árbol B^+.

5. CONCLUSIONES

En este trabajo se describieron dos métodos de almacenamiento para las bases de datos temporales utilizando técnicas espaciales. Se efectuó también la evaluación del rendimiento de estos métodos con un enfoque probabilístico.

El primer método de almacenamiento espacial presentado, el árbol R, puede ser utilizado para indexar una relación temporal. Los Rectángulos de los nodos no hojas engloban los rectángulos de sus hijos y las entradas de los nodos hojas apuntan hacia objetos cuyos rectángulos (segmentos) son iguales al intervalo de tiempo de las tuplas.

El Segundo método espacial, la indexación directa con árboles B^+, aprovecha los árboles B^+ existentes en los SGBD tradicionales para indexar las relaciones temporales. Esta técnica representa los intervalos de tiempo de las tuplas de una relación como puntos en un espacio bidimensional discreto y luego se definen diversos órdenes lineales sobre tales puntos. Por lo tanto, una selección temporal es tratada como una búsqueda espacial.

Como el árbol R es equilibrado, el costo de recuperar el objeto es igual a la altura del árbol. El inconveniente es que a veces es necesario recorrer varios caminos de la raíz del árbol a las hojas debido a los solapamientos de los rectángulos asociados a los objetos. Por tanto, el rendimiento de la operación de búsqueda se degrada.

En la evaluación del rendimiento se demostró que el método de indexación directa con árboles B^+ es mas eficaz que el Arbol R, tanto en costos de almacenamiento como en costos de consultas sobre un intervalo de tiempo.

La realización de este trabajo mostró la importancia de estudiar el sistema de gestión de bases de datos POSTGRES pues se obtuvo un conocimiento técnico que permitirá enriquecer este sistema con un nuevo método de acceso: La indexación directa de información temporal mediante el árbol B^+.

Además con la realización de los prototipos se dispondrá de un código fuente sobre el método de indexación directa que facilitará en un futuro la intervención del SGBD POSTGRES.

El estudio de los métodos de acceso del POSTGRES se dificultó debido a la falta de información sobre ciertas estructuras de datos. De todas maneras cuando se proceda a implementar el método de indexación directa de información temporal con el árbol B+ será necesario completar las especificaciones presentadas en este documento.

La indexación directa de información temporal mediante el árbol B^+ es un método que realiza transformaciones de tiempo de las relaciones temporales en un espacio bidimensional y luego obtiene un orden lineal en este plano. Ahora, teniendo en cuenta que esta indexación es más eficaz que el árbol R utilizado por el SGBD POSTGRES es posible enriquecer el optimizador de consultas combinando los ordenes VERTICAL, HORIZONTAL y DIAGONAL para construir dinámicamente los índices con los árboles B^+ existentes en el SGBD POSTGRES.

REFERENCIAS

[FSR87] C. Falutsos, T. Sellis y N. Roussopoulos. The R^+ - Tree: A Dynamic Index for Multidimensional Objects. In Proc. 13 th International Conference on VLDB, pág 507-518, England, September 1987.

[Glot96] C.H. Goh, H. Lμ, B.C. Oid and K.L. Tan. Indexing Temporal Data Using Existing B^+ trees. En Data & Knowledge Engineering 18, pág 147-165, 1996.

[Gut84] A. Guttman, R-trees: A Dynamic Index Structure for Searching. En Proceedings of the 1984 ACM-SIGMOD International Conference on Management of Data, Boston, MA, June 1984.

[GY88] S.K. Gadia y C.S. Yeung. A Generalized Model for a Relational Temporal Database. En proceedings of ACM SIGMOD International Conference on Management of Data, pág 251-259, Chicago Ill, June 1988.

[SA85] R. Snodgrass e I. Ahn. A Taxonomy of Time in Databases. En Proceedings of ACM SIGMOD International Conference on Management of Data, pág 236-246, Austin, TX, May 1985.

[Ston87] M. Stonebraker, The Design of the POSTGRES Storage System, Proc 13th VLDB Conferenca, pages 289-300. Brighton 1987.

[Ston90] M. Stonebraker, The implementation of POSTGRES, IEEE transaction on Knowledge and Data Engineering, pág. 125-142, March 1990.