Universidad de Buenos Aires
Facultad de Ingeniería
Departamento de Informática

Organización de Datos (75.06)

Voto Electrónico
Documentación Organizacional

Cuatrimestre y año: 2do Cuatrimestre 2011

Docente a cargo del TP: Nicolás Pablo Fernández Theillet

Grupo: Lamas

Fecha de Entrega: 2011-11-25

Integrantes:

Padrón	***Nombre***	***Email***
91187	Gonzalez Durand, Juan Manuel	jmanuel.gonzalez.durand@gmail.com
90762	Ostrowsky, Gabriel	gaby.ostro@gmail.com
90728	Schenkelman, Damián	damian.schenkelman@gmail.com
91045	Torrado, Alejandro	aletorrado@gmail.com
90884	Zamudio, Gonzalo	ahogadosderazon@gmail.com

Índice

Diseño

En esta sección se detallan las decisiones de diseño tomadas para la organización de los archivos de las diferentes entidades del problema. Para cada una de ellas se detalla:

- La organización que se utilizará para guardar los datos relacionados con la misma.
- Las razones por la que se decidió dicha organización.
- Características particulares dentro de la organización elegida (ej: función de dispersión para el Hash).
- Estructura de los registros utilizados (tanto en índices como para guardar datos).
- Estructura de los bloques utilizados.
- Una imagen o gráfico que sirve para ilustrar el diseño elegido.

Diseño General Árbol B+

Algunas características, comunes a todos los árboles B+ se detallan a continuación para evitar su reiteración:

Separadores

- Concatenación: No implementada.
- Búsqueda: Binaria.

Manejo Overflow

- Condición: Se considera que un nodo entra en overflow cuando al insertar o actualizar un registro, el tamaño total ocupado por datos (sin considerar metadata), es mayor al 90% del tamaño disponible para datos en el nodo. No se considera el 100%, para permitir mayor flexibilidad en las operaciones, ya que al ser registros de longitud variable no se tiene seguridad sobre el tamaño de los registros.
- Determinación registro medio: Se suma el tamaño de los registros del bloque en orden, considerando también al registro que causo el overflow en el caso de la inserción. Se considera al registro del medio como aquel que hace que el tamaño acumulado sea mayor a la mitad del espacio máximo disponible para datos (es decir, 45% del espacio disponible para datos).
- Política Split: En caso de overflow se separan los registros en dos nodos. Al nodo original se le sacan el registro del medio y todos los siguientes y se agregan a un nuevo nodo que pasa a ser hermano derecho del que entro en overflow. Adicionalmente, se agrega al padre (nodo interno) un registro nuevo de índice, con la clave del registro del medio.

Manejo Underflow

- Condición: Se considera que un nodo entra en underflow cuando eliminar un registro, el tamaño total ocupado por datos (sin considerar metadata), es menor al 45% del tamaño disponible para datos en el nodo. No se considera el underflow en operaciones de actualización. Considerar el 50%, permite mayor flexibilidad en las operaciones, ya que al ser registros de longitud variable no se tiene seguridad sobre el tamaño de los registros.
- Balanceo: Se suma el tamaño total ocupado por datos en ambos bloques. Si la suma es mayor al tamaño máximo de datos permitido en un bloque se balancea al nodo que entro en underflow con su hermano derecho. El balanceo mueve el primer nodo del hermano derecho al izquierdo, hasta que el tamaño ocupado en el nodo izquierdo sea mayor al derecho.
    - Caso particular: En caso que el nodo que entra en underflow sea el último (es el hijo que está más a la derecha) de su padre, se balancea a este con su hermano izquierdo hasta que el hermano izquierdo quede con carga mayor.

- Política de Split: Si al momento de intentar el balanceo, el tamaño total ocupado por datos es menor al tamaño del nodo (sin considerar a la metadata) se mergea al nodo en underflow con su hermano derecho. Todos los registros del hermano derecho pasan al hermano izquierdo. Adicionalmente, se remueve del padre al índice relacionado con el hermano derecho (que es la última clave del padre que es menor al primer registro del nodo derecho, no necesariamente debería ser igual). Adicionalmente, se marca al nodo derecho como nodo libre en el archivo de nodos libres.
    - Caso particular: En caso que el nodo que entra en underflow sea el último (es el hijo que está más a la derecha) de su padre, se mergea a este con su hermano izquierdo (se sigue haciendo lo mismo en el padre).

Esta lógica es la misma en todos los casos de manera de implementar solo un Árbol B+ que pueda ser usado para todas las entidades que corresponda.

Distrito

Organización

Archivo de bloque con Registros de datos de longitud variable (RLV) guardados en un Árbol B+.

Razones

- La cantidad de distritos (al menos en un escenario real) es fija, no aumenta. Solo se harán operaciones de consulta una vez que la carga inicial fue realizada.
- Otra opción podría ser una organización directa, ya que es un archivo de datos maestros sin variaciones, por lo que un cálculo correcto en el tamaño inicial del hash permitiría un acceso O(1) en la gran mayoría de los casos.

Características Particulares

- Tamaño nodo: 1024 B.
- Búsqueda separador: Lineal.

Registros

Se utilizara solo un registro de datos, de la forma:

- **R**(longitudRegistro, longitudDistrito, (distrito)i)

Bloques

- SSB(nivel, espacioLibre, punteroNodo, (Distrito)+)
- ISB(nivel, espacioLibre, punteroNodoHijoIzq, (Distrito, punteroNodoHijo)+)

Ilustración

A continuación se muestra el posible contenido de un bloque:

Nivel	Espacio Libre	Siguiente Nodo	Distrito 1			Distrito 2			Distrito 3			Distrito 4		
0	971	5	7	6	Chaco	11	10	Catamarca	9	8	Córdoba	8	7	Chubut

Tamaño ocupado por cada campo (en bytes):

2	4	4	2	1	6	2	1	10	2	1	8	2	1	7

Espacio libre: 1024 – 53 = 971 B.

Votante

Organización

Archivo de bloque con Registros de datos de longitud variable (RLV) guardados en un Hash de dispersión fija con zona de desborde. La zona de desborde será un archivo secuencial de RLV organizado por bloques para poder direccionar a otra cubeta.

Razones

- Ninguna funcionalidad de la aplicación requiere recorrer distritos secuencialmente una vez que encontramos a uno.
- Es necesario, para poder proveer una buena experiencia de usuario al electorado, poder acceder a la información de un votante en la menor cantidad de accesos a disco posible y de forma aleatoria.

- Una función de hashing, y tamaño de bloque (tanto de datos como de desborde) correctamente elegidos pueden hacer que en el momento más crítico (el día de las elecciones) la cantidad de lecturas necesarias para acceder a un votante sea baja. Veamos esto con un poco más de detalle:
    - Durante una elección, si un bloque se desborda debido al crecimiento de un registro, solamente se necesitaría de otro bloque para acceder a un votante. Es decir, que en la gran mayoría de los casos, como máximo se necesitarían 2 accesos a disco.
    - Una vez que las elecciones finalizan, se puede evaluar el estado del hash para determinar si una re-estructuración de mantenimiento es conveniente (en base a la cantidad de bloques con desborde).

Características Particulares

- Función de Hash: Se optó por la función hash módulo que consiste en este caso en: (DNI % Cantidad de Bloques totales).
- Tamaño Bucket/Bloque: 16 KB, seteado por configuración.
    - La cantidad de votantes configurada en el archivo de generación es 50000.
    - A partir del tamaño del tamaño promedio del bloque y que debemos usar el valor más cercano hacia arriba de 512 * 2^n.
- Factor empaquetamiento (registros ocupados / espacio disponible): 80%.
- Cantidad de bloques totales: 1303
    - Registros a almacenar: (50000 registros) / 80% = 62500 registros
    - Registros por bloque: (16 Kbytes/bloque) / (347 bytes/registro) = 48 registros/bloque
    - Bloques totales: (62500 registros) / (48 registros/bloque) = 1303 bloques
- El bloque se considera desbordado cuando se algún votante no pueda insertarse por tener tamaño mayor al espacio libre disponible o alguna actualización haga que el bloque se desborde.

Registros

Se utilizara solo un registro de datos, de la forma:

- Votante (longitudRegistro, (DNI)i, longitudNyA, NombreyApellido, clave, domicilio(longitudCalle, calle, nro), longitudDistrito, (distrito)ie, ((eleccion(fecha(año, mes, dia), longitudCargo, cargo))ie)*)

El campo de la longitud del registro es de 2 bytes. Los campos de control de cada campo son de 1 byte.

Tamaño promedio registro

Para calcular el tamaño promedio de un registro consideramos la siguiente longitud de sus campos.

- Control: 12 bytes.
- DNI: 4 bytes.
- Nombre y Apellido: En promedio consideramos 25 bytes.
- Clave: 4 bytes. La clave será 4 caracteres cualesquiera. Tomamos como ejemplo la clave bancaria.
- Calle: En promedio, 10 bytes.
- Numero: 2 bytes.
- Distrito: En promedio, 10 bytes.
- Fecha: 4 bytes.
- Cargo: En promedio, 10 bytes.
- Elecciones promedio: En el DNI hay 32 lugares para anotaciones. Consideramos un adicional del 25% ya que muchas veces se da la situación que estos lugares se llenan. Aproximadamente 20 elecciones como promedio, considerando que hay gente de todas las edades votando.

El tamaño promedio de un registro sería de 347 bytes.

Bloques
Los bloques del archivo de datos y de desborde serán de la forma:

- **B**(bloqueDesborde, espacioLibre, (Votante)*)

Elección

Organización
Archivo de bloques secuencial indexado organizado con un árbol B+ y RLV. Tendremos un índice secundario de elecciones por distrito, organizado también en un árbol B+ con RLV.

Razones
- En base a la funcionalidad del programa, tenemos la necesidad de realizar ABM y lecturas al azar.
- No encontramos una función de dispersión que pueda distribuir a los registros de una manera lo suficientemente homogénea como para que el un archivo de organización directa asegure una baja cantidad de desbordes, haciendo así que las operaciones puedan tener un costo alto. La característica balanceada del árbol B+, nos ayuda a mantener un orden razonable para las operaciones.
- Podríamos usar otro tipo de organización, por ejemplo un árbol B, ya que en realidad no necesitamos las características secuenciales del árbol B+. Sin embargo, por diferentes cuestiones (tiempo para desarrollo, robustez) preferimos usar un árbol B+, ya que sus capacidades de búsqueda secuencial si son requeridas para otras entidades.
- Para la generación de votos aleatorios, es de gran beneficio tener indexadas las elecciones por distrito. De esta forma se pueden obtener las mismas, para que el votante vote en ellas, del árbol de elecciones sin la necesidad de recorrerlo todo, para

Características Particulares
- Tamaño nodo: 16384 B. Como una elección puede tener una gran cantidad de nodos el tamaño de un registro puede ser muy grande. Para asegurar que varias elecciones pueden incluirse en un bloque elegimos este tamaño.
- Búsqueda separador: Lineal.

Registros
Usaremos dos tipos de registros diferentes, uno para ser usado en el sequence set y otro para el index set.

Sequence Set
SSR (longitudRegistro, año, mes, dia, longitudCargo, cargo, longitudDistrito, primerDistrito, punteroListaDistritos)

Long Registro	Año	Mes	Día	Long Cargo	Cargo	Long Distrito	Primer Distrito
29	1986	06	22	11	Intendente	8	Cordoba

En el sequence set guardaremos la lista de distritos completa, ya que los registros utilizados son de longitud variable y no se justificaría tenerlos almacenados en un archivo a parte en este caso. Además, esto es porque asumimos que la mayoría de las elecciones tendrán un distrito y solo algunas más que esa cantidad.

Index Set
ISR (longitudRegistro, año, mes, día, longitudCargo, cargo)

Long Registro	Año	Mes	Día	Long Cargo	Cargo
16	1986	06	22	11	Intendente

Árbol Índice

Usaremos dos registros, uno para ser usado en el sequence set, otro para el index set.

Sequence Set

ISSR (longitudRegistro, longitudDistrito, distrito)

Index Set

IISR (longitudRegistro, longitudDistrito, distrito, cantidadElecciones, (longitudIdEleccion, año, mes, dia, longitudCargo, cargo)+)

Bloques

Tendremos dos tipos de organizaciones en bloques:

Sequence Set

SSB (nivel, espacioLibre, siguienteNodo, **SSR**+)

Index Set

ISB (nivel, espacioLibre, punteroNodoHijoIzq, (**ISR**, punteroNodoHijo)+)

Indice por Distrito

- SSB(nivel, espacioLibre, punteroNodo, ISSR+)
- ISB(nivel, espacioLibre, punteroNodoHijoIzq. (IISR, punteroNodoHijo)+)

Lista

Organización

Archivo de bloques secuencial indexado organizado con un árbol B+ y RLV.

Razones

- En base a la funcionalidad del programa, tenemos la necesidad de realizar ABM y lecturas al azar.
- Necesitamos la capacidad de recorrer secuencialmente al archivo de listas para generar las elecciones. Esto se debe a que a partir del Votante obtenemos al distrito, con el distrito a las elecciones del distrito y con las elecciones obtenemos la primera lista de la elección y realizamos búsqueda secuencial para recuperar todas.

Características Particulares

- Tamaño Bucket/Bloque: 8192 B. Va a haber una gran cantidad de listas por elección y el tamaño de las listas (en promedio) es pequeño comparado con otras entidades, por lo que usar este tamaño de bloque permitirá reducir la cantidad de lecturas a disco sin causar demasiada fragmentación interna.
- Busqueda separador: Lineal.

Registros

Se utilizará un solo registro.

R (longitudRegistro, año, mes, día, longitudCargo, cargo, longitudNombre, nombre)

El campo de longitud de registro es de 2 bytes. Los campos de control son de 1 byte.

Tamaño del registro

- Control: 2 bytes
- Fecha: 4 bytes

- Cargo: 10 bytes (promedio)
- Nombre: 20 bytes (promedio)

Tamaño promedio del registro: 38 bytes

Ejemplo

Long Registro	Año	Mes	Día	Long Cargo	Cargo	Long Nombre	Nombre
24	1986	06	22	11	Intendente	7	Lista0

Tamaño del registro ejemplo: 24 bytes

Bloques

- SSB(nivel, espacioLibre, punteroNodo, R+)
- ISB(nivel, espacioLibre, punteroNodoHijoIzq (R, punteroNodoHijo)+)

Conteo

Organización

Archivo de bloques secuencial indexado organizado con un árbol B+ y RLV. Los conteos serán ordenados por su Id, la cual concatenara primero al campo de la elección.

Tendremos un índice secundario de conteos por distrito, organizado también en un árbol B+ con RLV.

Razones

- En base a la funcionalidad del programa, tenemos la necesidad de realizar ABM y lecturas al azar.
- Adicionalmente, para los informes de distrito y elección, es altamente beneficioso tener acceso secuencial a los registros, de manera de reducir la cantidad de lecturas de disco.
- Para los reportes por elección y distrito necesitamos acceso a los conteos por esos campos.
    - Los conteos dentro del árbol estarán ordenados en primer factor por elección, pudiendo acceder a los mismos de manera secuencial.
    - El índice de distrito nos permite acceder a los conteos sin tener que recorrer todo el árbol.

Características Particulares

- Tamaño nodo: 2048 B.
- Búsqueda separador: Lineal.

Registros

Árbol Datos

Usaremos dos registros, uno para ser usado en el sequence set, otro para el index set.

Sequence Set

SSR(longitudRegistro, año, mes, dia, longitudCargo, cargo, longitudNombre, nombre, longitudDistrito, distrito, cantidad)

L Reg	Año	Mes	Dia	L Cargo	Cargo	L Nombre	Nombre	L Dist	Dist	Cant
37	1986	06	22	11	Intendente	7	Lista0	8	Cordoba	500000

Index Set

ISR(longitudRegistro, año, mes, dia, longitudCargo, cargo, longitudNombre, nombre, longitudDistrito, distrito)

Árbol Índice

Usaremos dos registros, uno para ser usado en el sequence set, otro para el index set.

Sequence Set

ISSR (longitudRegistro, longitudDistrito, distrito)

Index Set

IISR (longitudRegistro, longitudDistrito, distrito, cantidadConteos, (longitudIdConteo, año, mes, dia, longitudCargo, cargo, longitudNombre, nombre, longitudDistrito, distrito)+)

Bloques

Árbol Datos

- SSB(nivel, espacioLibre, punteroNodo, SSR+)
- ISB(nivel, espacioLibre, punteroNodoHijoIzq, (ISR, punteroNodoHijo)+)

Indice por Distrito

- SSB(nivel, espacioLibre, punteroNodo, ISSR+)
- ISB(nivel, espacioLibre, punteroNodoHijoIzq. (IISR, punteroNodoHijo)+)

Cargo

Organización

Archivo de bloques con Registros de datos de longitud variable (RLV) guardados en un Hash de dispersión fija con zona de desborde. La zona de desborde será un archivo secuencial de RLV organizado por bloques para poder direccionar a otra cubeta.

Razones

- No se requiere un acceso secuencial, por ejemplo para algún reporte.
- El acceso necesario es de tipo aleatorio. El Hash reduce considerablemente el tipo de acceso de este tipo.
- Se trata de una entidad que tiende a ser muy estática, es decir sin cambios a lo largo del tipo, ya que para un cargo principal, los cargos secundarios no suelen variar. De esta forma la problemática de la reestructuración del Hash no es importante para este caso.

Características Particulares

- Función de Hash: (Suma de cada letra de la key en valor ascii) % (cantidad de bloques).
- Tamaño de bloque: 32 K (seteado por configuración)
- Tamaño de registro promedio: 80 B
- Factor de empaquetamiento: 80%.
- Cantidad de bloques totales: 40 bloques
    - Registros a almacenar: (1500 registros) / 80% = 1875 registros
    - Registros por bloque: (32 Kbytes/bloque) / (80 bytes/registro) = 410 registros/bloque
    - Bloques totales: (1875 registros) / (48 registros/bloque) = 40 bloques

Registros
Se utilizara un registro del tipo:

R (longitudRegistro, longitudCargoPpal, (cargoPpal)i, cantCargosSecundarios, (longitudCargoSec, cargoSec)*)

Bloques
Los bloques del archivo de datos y de desborde serán de la forma:

B (bloqueDesborde, espacioLibre, (registroCargo)*)

Ilustración
A continuación se muestra el posible contenido de un bloque:

Desborde	Espacio libre	Registro 1		
-1	36	(41)(11)Presidente(2)(16)VicePresidente(11)Intendente		

Candidato

Organización
Archivo de bloques secuencial indexado organizado con un árbol B+ y RLV.

Razones
- Necesitamos acceder secuencialmente a esta estructura de forma ordenada, en especial para la generación de reportes.
- El árbol B+ provee las soluciones de indexación necesarias, con la ventaja adicional de suplir el requerimiento de secuencialidad.

Características particulares
- Tamaño nodo: 1024 B.
- Búsqueda separador: Lineal

Registros
Se utilizaran registros de sequence set e index set equivalentes, ya que el índice del registro es el propio registro en su totalidad.

Sequence Set / Index Set
(longitudRegistro, longitudLista, lista, longitudVotante, votante, longitudCargo, cargo)

Bloques

Sequence Set
SSB (nivel, espacioLibre, siguienteNodo, SSR+)

Index Set
ISB (nivel, espacioLibre, punteroNodoHijoIzq, (ISR, punteroNodoHijo)+)