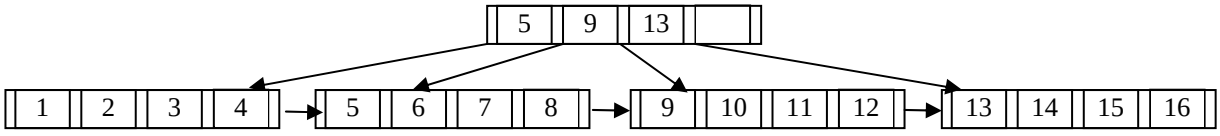
Examen de Ficheros y bases de datos (cód. 520)
Ingeniería Técnica en Informática de Gestión
Convocatoria de septiembre

II Parcial

1) (2,9 puntos).
   a) (0,8 puntos) Constrúyase un árbol B+ mínimo con *n*=5 para 16 valores diferentes de la clave, del 1 al 16.

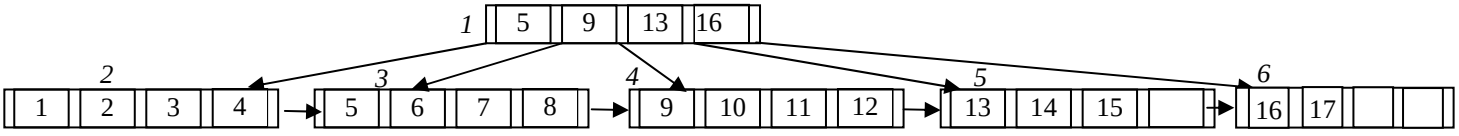
Solución:
Con *n*=5 se pueden alojar *n*-1=4 valores de la clave por nodo. Como todos los valores se deben representar en los nodos hoja, son necesarios $\left\lceil \frac{16}{4} \right\rceil = 4$ nodos hoja. Como restricciones que debe cumplir este árbol se tiene:
El nodo raíz tiene entre 1 y 5 hijos.
Cada nodo interno tiene entre $\left\lceil \frac{5}{2} \right\rceil = 3$ y 5 hijos.
Los nodos hoja contienen entre $\left\lceil \frac{5-1}{2} \right\rceil = 2$ y 5-1=4 valores.

En este caso, el nodo raíz puede apuntar a todos los nodos hoja porque *n* (número de punteros por nodo) es 4. No es posible estructuralmente usar un número menor de nodos. La estructura es:



   b) (0,8 puntos) Constrúyase el árbol resultado de añadir el valor 17.

Solución:
En este caso son necesarios $\left\lceil \frac{17}{4} \right\rceil = 5$ nodos hoja. Además hay que reorganizar para mantener la restricción del número mínimo de valores en los nodos hoja:.



   c) (1,3 puntos) Si se borrase este último valor insertado, ¿qué sería mejor desde el punto de vista del rendimiento: hacer mínimo el número de nodos o mantenerlos? Razonar en términos del número de accesos en lectura y escritura a bloques. Supóngase que cada nodo se implementa en un bloque, una

operación de escritura tiene un coste doble que una de lectura y no se tiene en cuenta el coste de los descartes de bloques.

Solución:

En ambos casos hay que localizar el valor a borrar, lo que implica dos lecturas de bloque.

a) Si se hace mínimo el número de nodos se llega a la situación del apartado b), y para ello hay que:
   1. Leer el bloque 1 para poder acceder a los bloques 5 y 6, y poder ajustarlo posteriormente.
   2. Leer el bloque *6* para descartarlo y ajustar los punteros del bloque *1*.
   3. Leer el bloque *5* para colocar el valor 16.
   4. Ajustar los punteros y valores del bloque *1*. Escribir el bloque *1*.
   5. Escribir el bloque *5*.
   6. Descartar el bloque *6*.

   Es decir, 3 lecturas y 2 escrituras = 3+2*2 = 7 unidades de coste.

b) Si se mantiene el número de nodos, el nodo hoja *6* se queda con menos valores de los que se permiten, por lo que hay que traer un valor del nodo hoja *5*, por ello hay que:
   1. Leer el bloque 1 para poder acceder a los bloques 5 y 6, y poder ajustarlo posteriormente.
   2. Leer el bloque *5* para trasladar el valor 15 al bloque *6*.
   3. Escribir el bloque *5*.
   4. Ajustar el valor del bloque *1* (ahora será 15 en lugar de 16). Escribir el bloque *1*.
   5. Leer el bloque *6*.
   6. Ajustar los valores del bloque *6* (ahora contendrá el 15 y el 16). Escribir el bloque *6*.

   Es decir, 3 lecturas y 3 escrituras = 3 + 3*2 = 9 unidades de coste.

Por lo tanto, es menos costoso mantener el número de nodos.

2) (1,3 puntos) Dado el siguiente código de Oracle:

```
SET AUTOCOMMIT OFF;
ACCEPT número PROMPT "Id: "
DELETE FROM tabla1 WHERE Id = %número%;
INSERT INTO tabla2 VALUES (%número%,'Nuevo valor');
COMMIT;
```

a) (0,3 puntos) ¿Qué significa la primera línea?

Solución:

Desactiva el autocompromiso de las instrucciones SQL de actualización.

b) (0,3 puntos) Si se ejecutan dos instancias de ese programa, ¿es posible el entrelazamiento de las instrucciones de modificación?

Solución:

No, porque las tablas se bloquean hasta que se alcanza el punto de compromiso.

c) (0,7 puntos) Sabiendo que el objeto del programa anterior es trasladar una tupla de tabla1 a tabla2, ¿qué situación anómala podría ocurrir si en la primera instrucción se escribe ON en lugar de OFF?

Solución:

Si ocurre un error entre las instrucciones DELETE e INSERT, como la interrupción de la transacción desde el sistema operativo, se habría eliminado una tupla de tabla1 que aún no se ha podido insertar en tabla2. Como el borrado está comprometido debido a la instrucción SET AUTOCOMMIT ON, el borrado es permanente y no se puede retroceder.

3) (1,6 puntos) Programar los disparadores que se estimen oportunos para actualizar la tabla Producto(A,B,C,D) cada vez que se modifique, inserte o borre cualquier tupla en las tablas Tabla1(A,B) y Tabla2(C,D). La tabla Producto debe contener el producto cartesiano de las tablas Tabla1 y Tabla2.

Solución:

La solución más inmediata es recalcular el producto cartesiano bajo cualquier modificación de las tablas Tabla1 y Tabla2, para lo que se necesitan dos disparadores, uno por cada tabla:

```
CREATE OR REPLACE TRIGGER ActualizaProductoTabla1
  AFTER INSERT OR DELETE OR UPDATE ON Tabla1
BEGIN
  DELETE * FROM Producto;
  INSERT INTO Producto SELECT * FROM Tabla1,Tabla2;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error inesperado.');
END;
/

CREATE OR REPLACE TRIGGER ActualizaProductoTabla2
  AFTER INSERT OR DELETE OR UPDATE ON Tabla2
BEGIN
  DELETE * FROM Producto;
  INSERT INTO Producto SELECT * FROM Tabla1,Tabla2;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error inesperado.');
END;
/
```

4) (4,2 puntos) Considérese un archivo r(A: Byte(2), B: Text(2.000.000), C: Date(4)), donde los tamaños de los tipos se dan en bytes, y con 1.048.576 filas. Se indexa bajo el campo A (con repeticiones) mediante una estructura de índice primario denso y asignación enlazada. Los bloques tienen un tamaño de 1.024 bytes, sus direcciones ocupan 4 bytes y disponen de mapas de bits de existencia para los registros. Se pide:
   a) (1,3 puntos) Describir la estructura y tamaño de la información almacenada en los ficheros de datos y de índice asumiendo una distribución uniforme de los valores del campo A.

Solución:

En media, el fichero de datos tendrá 16 entradas repetidas (1.048.576/65.536=16) por cada entrada del fichero de índice (que es denso y representa a todos los valores de la clave que aparecen en el fichero de datos). Por lo tanto, cada entrada del fichero de índice tiene apunta a la primera entrada (de 16 en total) del fichero de datos con valores iguales de la clave.

- Fichero de datos:
  Registros de tuplas (A, B, C)
  Tamaño: 1.048.576*(2+2.000.000+4)= 2.097.158.291.456 bytes, es decir, del orden de 2 TB.
  Bloque: Mapa de bits de existencia (N bits), N registros y dirección de enlace (4 bytes).
- Fichero de índice:

Registros de pares (Valor de la clave, Dirección de bloque)
Tamaño: 65.536*(2+4) = 393.216 bytes = 384 KB.
Bloque: Mapa de bits de existencia (N bits), N registros y dirección de enlace (4 bytes).

b) (1,3 puntos) Calcular el factor de bloqueo de los ficheros de datos y de índice.

Solución:

Para el fichero de datos:
$N*(2+2.000.000+4)*8+4*8+N \leq 1.024*8$, donde:

- $N*(2+2.000.000+4)*8 = N*16.000.048$ es el tamaño en bits ocupado por un registro (2 bytes para el campo A, 2.000.000 para el campo B y 4 para el campo C).
- $4*8 = 32$ es el número de bits ocupado por la dirección del siguiente bloque para la asignación enlazada
- N son los bits necesarios para el mapa de bits de existencia
- $1.024*8=8.196$ es el número de bits en los 1.024 bytes de un bloque

$N*(16.000.048+1) \leq 8.196-32$
$N \leq 0,00051025$
Por tanto, el factor de bloqueo N del fichero de datos es de 0,00051025 registros por bloque. Calculando su inversa, se obtiene que son necesarios 1959,8 bloques para alojar un registro. Por tanto, se usan 1960 bloques por cada registro de datos.

Para el fichero de índice:
$N*(2+4)*8+4*8+N \leq 1.024*8$, donde:

- $N*(2+4)*8=N*48$ es el tamaño en bits ocupado por un registro (2 bytes para el campo clave A y 4 bytes para la dirección de bloque).
- $4*8=32$ es el número de bits ocupado por la dirección del siguiente bloque para la asignación enlazada
- N son los bits necesarios para el mapa de bits de existencia
- $1.024*8=8.196$ es el número de bits en los 1.024 bytes de un bloque

$N*(48+1) \leq 8.196-32$
$N \leq 166,6$
Por tanto, el factor de bloqueo N del fichero de índice es de 166 registros por bloque.

c) (1,3 puntos) Calcular el número de accesos de entrada/salida en media para leer un registro bajo el valor de la clave y un valor del campo C, de forma que se determina unívocamente un registro en concreto de entre todos los posibles (es decir, el par <A,C> forma una clave candidata).

Solución:

Se deben recorrer 65.536/2=32.768 registros del fichero de índice secuencialmente, es decir, $\lceil 32.768/166 \rceil$= 198 bloques, para localizar la dirección del registro en el fichero de índice. Esta dirección apunta al bloque en el que se encuentra el primer registro del fichero de datos con el valor de búsqueda del campo A. Como hay 16 registros en media por cada valor de este campo, habrá que recorrer la mitad de ellos (también en media) para localizar el que contiene el valor de búsqueda del campo C. Por tanto, hay que leer 8 registros consecutivos del fichero de datos en media para localizar el registro buscado, lo que implica acceder a 1.960 bloques/registro * 8 registros = 15.680 bloques. En total se necesitan 197+1.960*8=15.877 operaciones de lectura de bloque.

d) (0,3 puntos) ¿Se mejoraría el tiempo de acceso si se añaden cajones para localizar los registros repetidos?

Solución:

No, porque en cualquiera de los casos habría que acceder a los registros de datos para conocer el valor del campo de búsqueda C. Lo que es peor, se añadiría un nuevo nivel de indirección que implicaría una operación extra de lectura del cajón, asumiendo que un cajón se implemente con un bloque.