Examen de Ficheros y bases de datos
Convocatoria de junio
II PARCIAL

1) (3 puntos) Considérese un fichero secuencial con un índice con asociación estática. El índice es primario y está organizado sobre una clave numérica sin repeticiones. Se dispone de asignación enlazada, mapas de bits de existencia en los bloques, un tamaño de bloque de 8.192 bytes y direcciones de bloque de 4 bytes. El registro de datos tiene dos campos: A (campo clave de 5 bytes) y B (300 caracteres) con codificación UNICODE y tamaño variable (necesita terminador de cadena). Se pide:
   a) (0,9 puntos) Calcular el factor de bloqueo de los cajones y del fichero de datos.

Solución:

Cajones (véase el apartado "Asociación estática (static hashing)" de los apuntes, pág. 6-24):
La estructura del índice: un campo para la clave (A, de 5 bytes), otro para la dirección de bloque correspondiente a la clave, mapa de bits de existencia y dirección de bloque por los cajones de desbordamiento. Por tanto, el factor de bloqueo N se calcula como:
$N \times (5+4) \times 8 + N + 4 \times 8 \leq 8.192 \times 8$
En esta inecuación se usa el bit como unidad y por ello todos los sumandos se multiplican por 8, salvo el mapa de bits, que ya va expresado en esta unidad.
$N \leq 897{,}31$
Por lo tanto, escogemos el mayor entero N menor o igual que 897,31, es decir, 897.

Fichero de datos:
La estructura del fichero de datos es: campos de datos (A, de 5 bytes, y B, de 300 caracteres = 300*2 bytes por ser UNICODE más dos bytes del terminador), mapa de bits de existencia y dirección de bloque por la asignación enlazada. Por tanto, el factor de bloqueo N se calcula como:
$N \times (5+(300+1)*2) \times 8 + N + 4 \times 8 \leq 8.192 \times 8$
$N \leq 13{,}49$
Por lo tanto, escogemos el entero N = 13.

   b) (0,9 puntos) Calcular el mínimo y el máximo espacio desperdiciado en bytes en un bloque del fichero de datos.

Solución:

El espacio mínimo desperdiciado corresponde a la situación en la que hay ocupación completa en el bloque. Por lo tanto, se calcula como el tamaño total del bloque menos el espacio ocupado por los 13 registros que puede contener y la información fija (mapa de bits y dirección de enlace).
$8.192 - (13 \times (5+(300+1)*2) \times 8 + 13 + 4 \times 8) = 295{,}375$ bytes, es decir, 295 bytes y $0{,}375 \times 8 = 3$ bits

El espacio máximo desperdiciado corresponde a la situación en la que hay sólo un registro en el bloque. Por lo tanto, se calcula como el tamaño total del bloque menos el espacio ocupado por el único registro y la información fija (mapa de bits y dirección de enlace).
$8.192 - (1 \times (5+(300+1)*2) \times 8 + 13 + 4 \times 8) = 7.579{,}375$ bytes, es decir, 7.579 bytes y $0{,}375 \times 8 = 3$ bits

   c) (0,6 puntos) Calcular el rango de la función de asociación para que con una distribución uniforme no existan cajones de desbordamiento.

Simplemente se divide el número posible de valores del campo clave entre el número de entradas que admite cada cajón (su factor de bloqueo, calculado en el primer apartado): 2^(5*8) / 1.149 = 956.929.180. Es decir, el rango de la función de asociación va de 0 a 956.929.179.

d) (0,6 puntos) Calcular la mejora de rendimiento cuando se usa el índice con respecto a que no se use al acceder al último registro del fichero de datos. Supóngase que éste contiene 16.000.000 de registros y que no hay cajones de desbordamiento.

Solución:

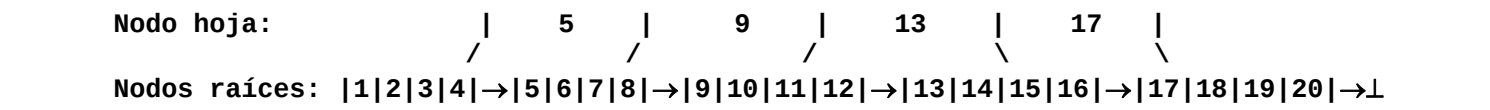
Al usar el índice, la función de asociación devuelve la dirección del cajón en el que se encuentra la dirección del bloque del fichero de datos en el que está el registro buscado. Por tanto, sólo 2 accesos.
Si no se usa el índice hay que recorrer secuencialmente todos los registros del fichero de datos. Como el factor de bloqueo es 13, es necesario leer 16.000.000 registros / 13 registros por bloque = 1.230.769,23 bloques. Hay que tomar el entero superior: 1.230.780 bloques. Por tanto, la mejora de rendimiento es de 1.230.780 / 2 = 615.390 ×
Es decir, ¡más de seiscientas mil veces más rápido!

2) (2,5 puntos) Árbol B+.
   a) (0,8 puntos) Constrúyase un árbol B+ con grado de salida (*n*) 5 que indexa los valores 1 a 20 de un campo numérico. Indíquense las restricciones que debe cumplir el árbol.

Solución:

Cada nodo interno (no hoja) tiene entre $\lceil n/2 \rceil$ y n hijos: 3-5.
El nodo raíz tiene entre 1 y n hijos: 1-5.
Los nodos hoja contienen entre $\lceil (n-1)/2 \rceil$ y n–1 valores: 2-4.

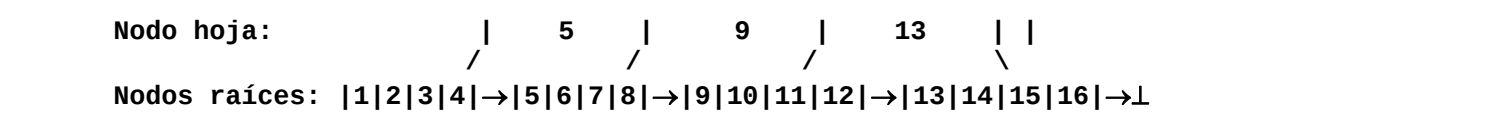
Para calcular el número mínimo de nodos hoja: 20 valores / 4 valores por nodo = 5 nodos.
Como un nodo raíz puede tener hasta 5 hijos, bastan dos niveles: nivel raíz y nivel hoja:



   b) (0,8 puntos) Determínese el número de accesos (sólo relativos al fichero de índice) necesarios para eliminar todos los valores de uno de los nodos hoja.

Solución:

Consideremos la eliminación de todos los valores del nodo hoja más a la derecha, es decir, la eliminación de los valores 17 al 20. El árbol queda:



Son necesarios:
- 2 accesos en lectura para alcanzar el nodo
- 1 acceso en escritura para modificar el penúltimo nodo hoja (cerrar la cadena de referencias)
- 1 acceso en escritura para modificar el nodo raíz

   c) (0,9 puntos) Calcúlese el tiempo de acceso al primer registro de datos realizando una búsqueda mediante clave. Asúmase que se dispone de un disco de 5.400 rpm con un tiempo de búsqueda de 7 ms y 256 sectores por pista, que cada nodo del árbol ocupa un bloque y que cada registro del fichero de datos también ocupa un bloque.

Solución:

Suponiendo una distribución aleatoria de los bloques en disco, el tiempo medio de lectura o de escritura de un bloque es:
Tiempo medio operación E/S = Tiempo de búsqueda + Tiempo de latencia + Tiempo de transmisión

Tiempo de búsqueda = 7 *ms*

Tiempo de latencia = 1/2*Inversa de la frecuencia = $\frac{1}{2}\frac{1}{5.400\frac{1}{\min}\frac{1\min}{60.000ms}} = 5{,}56ms$

Tiempo de transmisión= $\frac{1}{256}$ *Tiempo de rotación(el doble del tiempo de latencia)=0,04 *ms*
Tiempo medio operación de lectura de un bloque = 7+5,56+0,04 = 12,60 *ms*

El número de accesos necesarios es 3:
- 2 accesos debido a los 2 niveles
- 1 acceso para leer el registro de datos

El tiempo total de acceso es: 12,60 *ms* * 3 = 37,8 *ms*

3) (4,5 puntos) En la gestión de billetes de la red de transportes públicos de cierta comunidad española se emplea el siguiente modelo relacional:

Zonas (ID_Zona, Sencillo, Joven, Abono_Mensual, ID_AT)
↙
Abono_Transportes(ID_AT, Normal, Joven, TA_EDAD*, Anual*, Anual_TA_EDAD*)

donde ID_Zona y ID_AT son identificadores únicos de tipo VARCHAR2 (20) y el resto de atributos son precios en euros (NUMBER). Como regla, el importe del abono transporte 'Normal' es la suma de los importes de los abonos mensuales de las zonas que cubre.

a) (0,75 puntos) Escribe el correspondiente código DDL para la creación de estas tablas incluyendo las claves, restricciones y tipos señalados.

Solución:

```
CREATE TABLE "ABONO_TRANSPORTES"
  (
    "ID_AT"         VARCHAR2(20 BYTE),
    "NORMAL"        NUMBER NOT NULL ,
    "JOVEN"         NUMBER NOT NULL ,
    "TA_EDAD"       NUMBER,
    "ANUAL_NORMAL"  NUMBER,
    "ANUAL_TA_EDAD" NUMBER,
    CONSTRAINT "ABONO_TRANSPORTES_PK" PRIMARY KEY ("ID_AT")
  )

CREATE TABLE "ADMINUSER"."ZONA"
  (
    "ID_ZONA"         VARCHAR2(20 BYTE),
    "PRECIO_SENCILLO" NUMBER(5,0) NOT NULL,
    "PRECIO_BONOTREN" NUMBER(5,0) NOT NULL,
    "ABONO_MENSUAL"   NUMBER(5,0) NOT NULL,
    "ID_AT"           VARCHAR2(20 BYTE),
    CONSTRAINT "ZONA_PK" PRIMARY KEY ("ID_ZONA"),
    CONSTRAINT "AT_FK" FOREIGN KEY ("ID_AT")
        REFERENCES "ABONO_TRANSPORTES" ("ID_AT")
  )
```

b) (2 puntos) Teniendo en cuenta que el importe del abono transporte 'Normal' es la suma de los importes de los abonos mensuales de las zonas que cubre (ocurre igual con el 'Joven', solo que se le aplica un descuento), codifica un TRIGGER que se dispare tras:
   a. Insertar una fila en la tabla Zonas. Aquí deberá comprobar si existe la clave ID_AT en la tabla Abono_Transportes. Si la clave ya existe entonces actualizará el importe del los precios 'Normal' y 'Joven' (aplicando un factor de descuento 0,55) sumándole el importe del 'Abono Mensual'. Si la clave no existe deberá crearla dando el importe del 'Abono Mensual' al abono transporte 'Normal' y al 'Joven', éste último con el mencionado descuento.
   b. Actualizar el precio de un abono mensual: se actualizarán los importes del abono transporte 'Normal' y del 'Joven', éste último con el mismo descuento.

Solución:

```
CREATE OR REPLACE TRIGGER TRIGGER1 AFTER
  INSERT OR
  UPDATE OF ABONO_MENSUAL ON ZONA
  FOR EACH ROW

  DECLARE
  existe_clave NUMBER;

  BEGIN
    IF INSERTING THEN
      --Cursor para manejar la existencia de una clave en Abono_Transportes
      SELECT COUNT (*)
      INTO existe_clave
      FROM ABONO_TRANSPORTES
      WHERE ID_AT=:NEW.ID_AT;

      --Si existe actualizo el Abono Transporte según el criterio del
enunciado
      IF existe_clave = 1 THEN
        UPDATE ABONO_TRANSPORTES
        SET ABONO_TRANSPORTES.NORMAL=(ABONO_TRANSPORTES.NORMAL +
:NEW.ABONO_MENSUAL),
          ABONO_TRANSPORTES.JOVEN   =(ABONO_TRANSPORTES.JOVEN  +
:NEW.ABONO_MENSUAL*0.55)
        WHERE ID_AT                 =:NEW.ID_AT;

       --Si no existe el Abono Transporte, lo creamos con el mismo criterio

      ELSE
        INSERT
        INTO ABONO_TRANSPORTES VALUES
          (
            :NEW.ID_AT,
            :NEW.ABONO_MENSUAL,
            :NEW.ABONO_MENSUAL*0.55,
            NULL,
            NULL,
            NULL
          );
      END IF;

      --Si se actualiza el atributo Abono Mensual de una zona, recalculamos
      el de los Abonos Transportes 'Normal' y ' Joven'

    ELSIF UPDATING THEN
      UPDATE ABONO_TRANSPORTES
```

```
      SET ABONO_TRANSPORTES.NORMAL=(ABONO_TRANSPORTES.NORMAL -
:OLD.ABONO_MENSUAL + :NEW.ABONO_MENSUAL),
        ABONO_TRANSPORTES.JOVEN   = (ABONO_TRANSPORTES.JOVEN -
:OLD.ABONO_MENSUAL + :NEW.ABONO_MENSUAL*0.55)
      WHERE ID_AT                 =:NEW.ID_AT;
    END IF;
  END;
```

c) (1,75 puntos) Crear un procedimiento que recibe como parámetro de entrada un identificador de abono transporte (PARAM_IDAT) y que devuelve como salida:
   a. El identificador de las zonas que cubre y el importe del abono mensual 'Normal' para cada una de ellas.
   b. El número de zonas que cubre y la suma de los abonos mensuales 'Normal'
   c. El importe del abono transporte Normal que tiene asociado estas zonas (debería coincidir con el anterior).
   d. Si el parámetro de entrada hace referencia a un ID_AT inexistente, deberá controlarse mediante una excepción e informar al usuario por pantalla.

Solución:

```
CREATE OR REPLACE
PROCEDURE PROCEDURE1
(
  PARAM_IDAT IN VARCHAR2
) AS


var1 NUMBER;
contador INTEGER;
fila_AbonoT ABONO_TRANSPORTES%ROWTYPE;

-- Cursor que maneja el listado de Zonas ( y el importe de su Abono Mensual)
-- que comparten un Identificador de AbonoTransporte

CURSOR c_Zona IS
SELECT ID_ZONA, ABONO_MENSUAL
FROM ZONA
WHERE ID_AT=PARAM_IDAT;

Excep_NoZona EXCEPTION;

BEGIN


SELECT * INTO fila_AbonoT
FROM ABONO_TRANSPORTES
WHERE ID_AT=PARAM_IDAT;

 var1:=0;
 contador:= 0;

 dbms_output.put_line ('### RESULTADO DEL PROCEDIMIENTO ###');


-- Uso de Cursor con FOR


    FOR z IN c_Zona LOOP
     dbms_output.put_line('Zona: '|| z.ID_ZONA);
```

```
     dbms_output.put_line('Abono Mensual: '||z.ABONO_MENSUAL);

     var1:= var1 + z.ABONO_MENSUAL;
     contador:= contador+1;

    END LOOP;


    IF contador = 0 THEN
    raise Excep_NoZOna;

    ELSE
    dbms_output.put_line ('El precio de Abono Transporte Normal para estas
'|| contador ||' Zonas debería ser:'|| var1);
    dbms_output.put_line ('El precio de Abono Transporte Normal para estas
Zonas:'|| fila_AbonoT.NORMAL);
    END IF;

EXCEPTION

WHEN Excep_NoZona THEN
dbms_output.put_line ('El ABONO TRANSPORTE '|| PARAM_IDAT ||' NO EXISTE' );

END;
```