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

1) (4,3 puntos) Considérese un fichero secuencial indexado con un índice numérico (con valores entre 0 y 65.535) con duplicados y 250.000 registros. Se dispone de asignación enlazada, mapas de bits de existencia en los bloques, un tamaño de bloque de 4.096 bytes y direcciones de bloque de 4 bytes. El registro de datos tiene tres campos: A (campo clave de 2 bytes), B (100 caracteres) y C (500 caracteres) con codificación ASCII y tamaño variable (necesita terminador de cadena). Se pide:
   a) (1 punto) Calcular el tamaño del fichero de datos y del fichero de índice teniendo en cuenta sólo los tamaños de los registros. Si la distribución de valores de la clave es uniforme, ¿cuántos registros de datos hay por cada valor de la clave? Si en lugar de ser clave el campo A lo fuese el C, ¿por cuánto se multiplicaría el tamaño del fichero de índice?

**Solución:**

Tamaño fichero de datos:
Registro: 2 (A) + (100+1) (B) + (500+1) (C) = 604 bytes
Fichero: 250.000*604 = 151.000.000 bytes (144 Mb aprox.)
Tamaño fichero de índice:
Registro: 2 (A) + 4 (Dirección) = 6 bytes
Fichero: 65.536*6 = 1.500.000 bytes (1,43 Mb aprox.)
Registros de datos por valor de la clave:
250.000/(2^16) = 3,81 en media
Tamaño fichero de índice con campo clave C:
Registro: 501 (A) + 4 (Dirección) = 505 bytes
Fichero: 250.000*505 = 126.250.000 bytes (120 Mb aprox.)
Proporción entre los dos casos (clave A y clave C):
126.250.000/1.500.0000 = 84,17 ×

   b) (1 punto) Calcular el factor de bloqueo del fichero de índice bajo el campo A y del fichero de datos.

**Solución:**

La estructura del índice: un campo para la clave (A, de 2 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:
$N \times (2+4) \times 8 + N + 4 \times 8 \leq 4.096 \times 8$
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.
$N \leq 668,08$
Por lo tanto, escogemos el mayor entero N menor o igual que 668,08, es decir, 668.

Fichero de datos:
La estructura del fichero de datos es: campos de datos:
A, de 2 bytes, B, de 101 caracteres = (100 bytes más 1 byte del terminador), y C (501)
mapa de bits de existencia y dirección de bloque por la asignación enlazada. Por tanto, el factor de bloqueo N se calcula como:
$N \times (2+(100+1)+(500+1)) \times 8 + N + 4 \times 8 \leq 4.096 \times 8$
$N \leq 6,77$
Por lo tanto, escogemos el entero N = 6.

c) (1 punto) 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).
4.096 × 8 – (6 × (2+(100+1)+(500+1)) × 8 + 6 + 4 × 8) = 4.807 bits = 467,25 bytes, es decir, 467 bytes y 0,25 × 8 = 2 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).
4.096 × 8 – (1 × (2+(100+1)+(500+1)) × 8 + 6 + 4 × 8) = 29.030 bits = 3.487,25 bytes, es decir, 3.487 bytes y 0,25 × 8 = 2 bits

d) (1,3 puntos) Calcular el tiempo necesario para leer todos los registros de datos correspondientes al valor de la clave 2.048 para un disco de 5.400 rpm con un tiempo de búsqueda de 6 ms, 256 sectores por pista y 1.024 bytes/sector. Los sectores de un bloque no tienen por qué ser contiguos.

**Solución:**

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

Tiempo de búsqueda = 6 *ms*

Tiempo de latencia = 1/2*Inversa de la frecuencia = $\frac{1}{2}\frac{1}{5.400\frac{1}{\min}\frac{1\min}{60.000ms}} = 5{,}55ms$

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 = 6 + 5,55+0,04 = 11,60 *ms*

El número de accesos necesarios al índice se calcula dividiendo el número de registros a leer entre el factor de bloqueo del fichero de índice: (2048+1)/668=3,06 => 4 accesos (está en el cuarto bloque). Además se debe sumar los accesos debido a los registros del fichero de datos (número de registros por cada valor de la clave dividido entre su factor de bloqueo): 3,81/6 = 0,635 => 1 acceso (sólo hay que leer un bloque).
Además hay que considerar que un bloque ocupa cuatro sectores (un bloque es de 4.096 bytes mientras que un sector es de 1.024: ($\frac{4.096bytes/bloque}{1.024bytes/\text{sec}tor}$).). Así que para calcular el tiempo para acceder a un bloque se debe multiplicar el tiempo de acceso a un sector por cuatro.

El tiempo total de acceso es, por tanto: 5*4*11,60 *ms* = 232 *ms*

2) (2,5 puntos) Considérese un árbol B+ con grado de salida *n* = 3 que indexa los valores 1 a 12 de un campo numérico.
   a) (0,5 puntos) Indíquense las restricciones que debe cumplir el árbol.

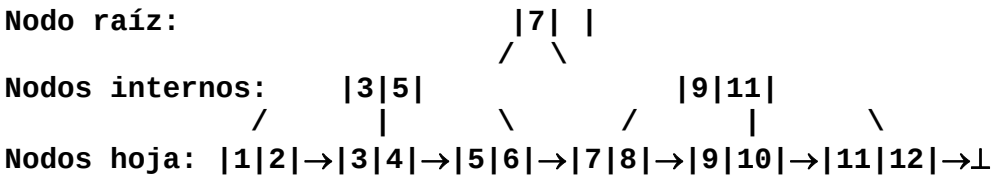
**Solución:**

Cada nodo interno (no hoja) tiene entre $\lceil n/2 \rceil$ y n hijos: 2-3.
El nodo raíz tiene entre 1 y n hijos: 1-3.

Los nodos hoja contienen entre $\lceil (n-1)/2 \rceil$ y n–1 valores: 1-2.

b) (1 punto) Constrúyase el árbol con el mínimo número de nodos.

Para calcular el número mínimo de nodos hoja: 12 valores / 2 valores por nodo = 6 nodos.
Es necesario contar con dos nodos internos para acceder en total a estos 6 nodos hoja, y un nodo raíz con dos hijos a los internos.:




c) (1 punto) Si cada nodo se almacena en un bloque de 128 bytes, las direcciones de bloque ocupan 4 bytes y los valores de la clave ocupan 32 bytes, ¿cuál es el espacio desperdiciado en total? Considérese que la ausencia de un valor de clave en un bloque se representa por su puntero a la derecha con un valor al propio bloque.

**Solución:**

En cada nodo se necesitan 3*4 bytes para las direcciones más 2*32 bytes para los datos, en total 76 bytes. Como cada bloque tiene 128 bytes, en cada nodo se desperdician 128-76 = 52 bytes. Además en el nodo raíz se queda un hueco de 32 bytes. El espacio total desperdiciado es 9*52+32 = 500 bytes (hay 9 nodos en total en el árbol).

3) (1,6 puntos) Escríbase una transacción como un procedimiento almacenado PL/SQL que transfiera fondos por cantidad de 50 € del código de cuenta 'C1' al 'C2'. Asúmase una tabla **`cuentas(código_cuenta varchar(20), saldo number(10,2))`**. Si el saldo de la cuenta destino fuese superior a 10.000€ después de la transferencia, añadir una bonificación de 5€. Si el saldo de la cuenta origen resultase negativo al realizar esta transferencia, añadir un préstamo a la tabla **`préstamos(código_préstamo varchar(20), código_cuenta varchar(20), importe number(10,2))`** por el valor absoluto de ese saldo negativo, y dejar el saldo de la cuenta a 0.

**Solución:**

```
create sequence seq_préstamos increment by 1 start with 1 nomaxvalue;

create table cuentas(código_cuenta varchar(20), saldo number(10,2));
create table préstamos(código_préstamo varchar(20), código_cuenta
varchar(20), importe number(10,2));

insert into cuentas values ('C1',75);
insert into cuentas values ('C2',100);

set transaction isolation level read committed;
set autocommit off;

create or replace
procedure traspaso as
  v_saldo cuentas.saldo%TYPE;
begin
  update cuentas set saldo = saldo + 50 where código_cuenta='C2';
  update cuentas set saldo = saldo - 50 where código_cuenta='C1';
  select saldo into v_saldo from cuentas where código_cuenta='C1';
  if (v_saldo<0) then
    update cuentas set saldo = 0 where código_cuenta='C1';
```

```
    insert into préstamos values (seq_préstamos.nextval,'C1',v_saldo);
  end if;
  select saldo into v_saldo from cuentas where código_cuenta='C2';
  if (v_saldo>10000) then
    update cuentas set saldo = saldo+5 where código_cuenta='C2';
  end if;
  commit;
end;

select * from cuentas;

C1   75
C2   100

call traspaso();
select * from cuentas;
C1   25
C2   150
call traspaso();
select * from cuentas;
C1   0
C2   200
select * from préstamos;
1    C1   25
call traspaso();
select * from cuentas;
C1   0
C2   250
select * from préstamos;
1    C1   25
2    C1   50
update cuentas set saldo=9990 where código_cuenta='C2';
call traspaso();
select * from cuentas;
C1   0
C2   10045
select * from préstamos;
1    C1   25
2    C1   50
3    C1   50
```

4) (1,6 puntos) Escríbase un disparador en PL/SQL que actualice el número total de butacas vendidas por cada espectáculo que se registra en la tabla **`resumen_ventas(evento varchar(20), número_vendidas int)`** a partir de las inserciones o borrados sobre la tabla **`reservas(evento varchar(20), fila varchar(20), columna varchar(20))`**. Se asume que ya hay un registro en la tabla **`resumen_ventas`** por cada evento que pueda aparecer en **`reservas`**, y que la modificación en **`reservas`** es consistente (por ejemplo, la aplicación ya se ha encargado de no reservar la misma butaca dos veces).

**Solución:**

```
create table resumen_ventas(evento varchar(20), número_vendidas int);
create table reservas(evento varchar(20), fila varchar(20), columna
varchar(20));

insert into resumen_ventas values('Opera',0);

CREATE OR REPLACE
```

```
TRIGGER Actualiza_resumen AFTER
  INSERT OR DELETE ON reservas
  FOR EACH ROW
  DECLARE
  BEGIN
    IF INSERTING THEN
      UPDATE resumen_ventas
        SET número_vendidas=número_vendidas+1
        WHERE resumen_ventas.evento = :NEW.evento;
    ELSIF DELETING THEN
      UPDATE resumen_ventas
        SET número_vendidas=número_vendidas-1
        WHERE resumen_ventas.evento = :OLD.evento;
    END IF;
  END;
```

Prueba del disparador:

```
insert into reservas values('Opera','1','1');
select * from resumen_ventas;
delete from reservas;
select * from resumen_ventas;
```