Ficheros y bases de datos
Ingenierías Técnicas en Informática
Convocatoria final de junio
II Parcial

1. (2,5 puntos) Un fichero de datos con campos A: Byte, B: Char(4.000) con 300.000 filas está indexado bajo el atributo A mediante una estructura de índice primario denso y asignación enlazada. Se asume que el campo A tiene valores comprendidos entre 0 y 255 distribuidos uniformemente y el campo B tiene codificación ASCII. Suponiendo que las direcciones de bloques de disco ocupan 6 bytes y tienen mapas de bits de existencia para los registros. Se pide:
   a) (0,5 puntos) ¿Con qué tamaño de bloque, en bytes, se debe dar formato al disco para asegurar la menor fragmentación de memoria para el fichero de datos con un factor de bloqueo de 4?

Factor de bloqueo:4
(1+4.000)*4*8+4+6*8 ≤ B*8
Primer sumando:
(1+4.000): Tamaño del registro en bytes
(1+4.000)*4: Número de bytes por los 4 registros en el bloque
(1+4.000)*4*8: Número de bits por los 4 registros en el bloque
Segundo sumando:
4: 4 bits necesarios para el mapa de bits de existencia
Tercer sumando:
6: Tamaño en bytes de la dirección del siguiente bloque en la asignación enlazada
6*8: Tamaño en bits de la dirección del siguiente bloque en la asignación enlazada
Lado derecho de la inecuación:
B: Tamaño del bloque en bytes
B*8: Tamaño del bloque en bits
Resolviendo:
B ≥ 16.010,5
Por lo tanto, B = 16.011 bytes

   b) (0,5 puntos) Determínese el espacio desperdiciado en cada bloque y en todo el fichero de datos.

El espacio desperdiciado en cada bloque es:
B*8-((1+4.000)*4*8+4+6*8)= 4 bits desperdiciados por bloque
Donde:
B*8 es el tamaño del bloque en bits.
(1+4.000)*4*8+4+6*8 es el tamaño de los cuatro registros en el bloque, además del mapa de existencia y la dirección de enlace.

Para calcular el espacio desperdiciado en todo el fichero de datos hay que calcular cuántos bloques ocupa. Dado que el factor de bloqueo es 4:

$$B_{Datos}=300.000/4=75.000 \text{ bloques}$$

Como en cada bloque se desperdician 4 bits, el espacio total desperdiciado resulta:
75.000*4bits=37.500 bytes

   c) (0,5 puntos) Si se exigiese que el tamaño de bloque fuese potencia de dos, ¿cambiaría el factor de bloqueo? ¿Cuál sería el espacio desperdiciado en cada bloque?

El problema se reduce a buscar la menor potencia de dos que sea mayor o igual que el tamaño de bloque calculado:

$$2^n \geq 16.011 \quad n \geq \log_2 16.011 = 13,9667758 \rightarrow n=14 \quad B=2^n=16.384$$

Se desperdician 16.384-16.010,5= 373,5 bytes, que no son suficientes para alojar un nuevo registro que ocupa más de 4.000 bytes. Por lo tanto, no cambia el factor de bloqueo.
Recordatorio:

$$\log_b(x) = n \Leftrightarrow x = b^n \rightarrow \log_2 = \ln(x)/\ln(2)$$

   d) (0,5 puntos) Calcúlese el factor de bloqueo para el fichero de índices y el espacio desperdiciado en este caso.

Para calcular el factor de bloqueo del fichero de índices:

$$(1+6)*N*8+N+6*8 \leq 16.011*8$$

Primer sumando:
- **(1+6): Tamaño del registro en bytes**
- **(1+6)*N: Número de bytes por los N registros en el bloque (N es el factor de bloqueo)**
- **(1+6)*N*8: Número de bits por los N registros en el bloque**

Segundo sumando:
- **N: N bits necesarios para el mapa de bits de existencia**

Tercer sumando:
- **6: Tamaño en bytes de la dirección del siguiente bloque en la asignación enlazada**
- **6*8: Tamaño en bits de la dirección del siguiente bloque en la asignación enlazada**

Lado derecho de la inecuación:
- **B: Tamaño del bloque en bytes**
- **16.011*8: Tamaño del bloque en bits**

Resolviendo:

$$N \leq (16.011*8-6*8)/(7*8+1)=2.246,31 \rightarrow N=2.247$$

Espacio desperdiciado en el fichero de índices:
Como hay 256 registros en el fichero de índices que corresponden a todos los valores posibles del campo A y el factor de bloqueo es 2.247, sólo es necesario un bloque para albergarlos, y sobra bastante espacio:
- **B*8-((1+6)*256*8+N+6*8)**
- **16.011*8-((1+6)* 256*8+2.247+6*8)=116.047 bits =14.505,875 bytes = 14.505 bytes + 7 bits**

(Los bits de la parte fraccionaria de los bytes se calculan: 0,875*8=7)

e) (0,5 puntos) Asumiendo que el fichero de datos está compactado (no hay huecos en los bloques de este fichero) y una distribución uniforme de registros por valores clave, calcúlese el número de accesos de entrada/salida necesarios para obtener todos los registros con el valor de la clave 128 en dos casos: usando el índice y sin usarlo. Calcular el factor de ganancia del primero con respecto al segundo.

Usando el índice:
1 acceso para leer el índice+300.000/256/4 = 294 accesos en lectura, donde:
300.000./256=1171,875 son los registros que hay en el fichero de datos con un valor de la clave en concreto, considerando una distribución uniforme
1.171,875/4 son los bloques necesarios para albergar estos registros para el factor de bloqueo 4

Sin usar el índice:
Hay que recorrer secuencialmente el fichero de datos hasta encontrar el primer registro con la clave 128 y parar al encontrar registros con la siguiente clave (dado que el fichero de datos está ordenado por esta clave). Por tanto, hay que recorrer las claves 0-128, leyendo el siguiente número de registros:129*1171,875=151.171,875, que en bloques resulta: 151.171,875/4=37.792,9688=37.793 accesos en lectura

El factor de ganancia al usar el índice es: 37.793/294=128,5 ×

2. (1,5 puntos) Considérese un árbol B+ con n=6 (número máximo de hijos por nodo). Se pide:
   a) (0,5 puntos) Calcúlense las restricciones del árbol sobre: número de hijos del nodo raíz, número de hijos de los nodos internos y número de valores de los nodos hoja.

El nodo raíz tiene entre 1 y *n* hijos: 1 – 6.
Cada nodo interno (no hoja) tiene entre $\lceil n/2 \rceil$ y *n* hijos: 3 – 6.
Los nodos hoja contienen entre $\lceil (n-1)/2 \rceil$ y *n* – 1 valores: 3 – 5.

b) (0,5 puntos) Dada la secuencia de números del 1 al 25, constrúyase un árbol B+ con el mínimo número de nodos. Calcúlese el número de niveles de este índice. ¿Cuántos accesos son necesarios para leer el primer registro de datos según un valor de la clave de búsqueda?

Como en cada nodo hoja caben 5 valores: 25/5=5 nodos hoja. Además, como el raíz puede tener hasta 6 hijos, el árbol puede tener tan sólo 2 niveles. Los valores de los nodos hoja se ubicarían en orden creciente desde la izquierda hasta la derecha desde el primer nodo. Los

valores del nodo hoja serían: 6, 11, 16 y 21. Se necesitan 3 accesos para leer el primer registro de datos (dos del índice más uno del propio registro de datos).
Los nodos hoja contendrían:
|1,2,3,4,5|6,7,8,9,10|11,12,13,14,15|16,17,18,19,20|21,22,23,24,25|

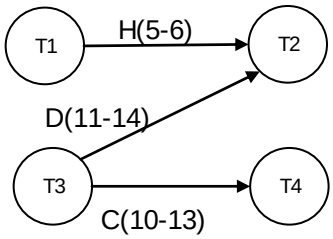
c) (0,5 puntos) Insértese el valor 26 en el árbol anterior y muéstrese el resultado. ¿Cuál es el número de niveles en este caso?

Se necesitan al menos $\lceil 26/5 \rceil$=6 nodos hoja, como cada hoja debe tener al menos 3 valores: 26/3=8 nodos a lo sumo. Tomando el valor menor (6) y dado que el raíz puede albergar 6 punteros, la altura del árbol no cambia.

3. (1,5 puntos) Dada la siguiente planificación:

	T1	T2	T3	T4
1	LOCK A			
2	LOCK E			
3	UNLOCK E			
4	LOCK H			
5	UNLOCK H			
6		LOCK H		
7			LOCK D	
8		LOCK B		
9			LOCK C	
10			UNLOCK C	
11			UNLOCK D	
12		UNLOCK B		
13				LOCK C
14		LOCK D		
15		UNLOCK D		
17	UNLOCK A			
18		UNLOCK H		
20				UNLOCK C

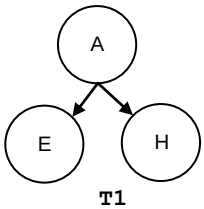
a) (0,75 puntos) Determinar si es secuenciable y, si lo es, determínese una planificación secuencial equivalente.



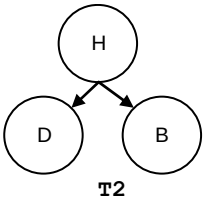
No se identifican ciclos en el grafo de precedencia, podemos afirmar que es secuenciable. Obtenemos una planificación secuencial equivalente aplicando ordenación topológica (aplicando el algoritmo sabemos que cualquier ordenación topológica empezará por T1 o T3)
T1→T3→T2→T4

b) (0,75 puntos) Diseñe el grafo con los valores A, B, C, D, E, H de la planificación anterior para asegurar que las transacciones T1, T2, T3 y T4 cumplen con el protocolo de árbol.

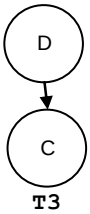
Para que cada transacción cumpla con el protocolo de árbol, sus respectivos grafos de datos deben ser (T4 sólo tendría el nodo C):



T1

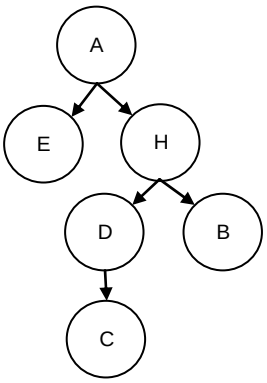


T2



T3

Uniendo los tres grafos consistentemente:



4. (4,5 puntos) La información de una BBDD de una aplicación que gestiona una "Web Social" es la siguiente:

- Los usuarios deben tener un identificador numérico único, nombre de usuario (no se podrá repetir), contraseña y dirección de correo electrónico. Los datos personales de los usuarios serán nombre, apellidos y fecha de nacimiento. En todo momento, cada usuario tendrá un "estado actual" de entre los siguientes: Conectado, Ausente, No disponible. El estado por defecto es "Ausente". También se almacenará el instante de última conexión.
- El portal permite subir fotos con texto asociado. A cada foto se le asigna un identificador único de manera automática. Es necesario almacenar este dato, así como un título para la foto, la fecha y hora de subida, el usuario que subió la foto (propietario), la visibilidad de la misma (pública, amigos ó privada), valoración (entre 0 y 10) y la ruta en que se almacena en el servidor. Cada foto podrá tener comentarios asociados. Cada uno de los comentarios tendrá texto (100 caracteres máximo), fecha y hora, y usuario que lo generó.
  a) (1 punto) Dado el siguiente esquema relacional. Codifica las sentencias de creación de tablas correspondientes. Elige los tipos de datos y añade las restricciones para los campos que estimes más convenientes. NOTA: No borrar en cascada en las restricciones de integridad referencial.

```
USUARIOS(idusu, nombre, apes, fecha_nac, email, usuario, password, estado, ultima_conex)
FOTOS(idfoto, titulo, fecha_subida, usuario, visibilidad, valoracion, ruta)
COMENT_FOTOS(idcom, comentario, fecha, usuario, foto)
```

```
DROP TABLE COMENT_FOTOS;
DROP TABLE FOTOS;
DROP TABLE USUARIOS;

CREATE TABLE USUARIOS (
  "IDUSU" INTEGER NOT NULL,
  "NOMBRE" VARCHAR(20),
  "APES" VARCHAR2(20),
  "FECHA_NAC" DATE,
  "EMAIL" VARCHAR2(40),
  "USUARIO" VARCHAR2(20),
  "PASSWORD" VARCHAR2(20),
  "ESTADO" VARCHAR2(20) DEFAULT 'Ausente' NOT NULL,
  "ULTIMA_CONEX" DATE,
  CONSTRAINT "USUARIOS_PK" PRIMARY KEY ("IDUSU"),
  CONSTRAINT "USUARIO_UNICO" UNIQUE ("USUARIO"),
  CONSTRAINT "ESTADO" CHECK ("ESTADO" IN ('Conectado','Ausente','No disponible'))
);
CREATE TABLE FOTOS (
  "IDFOTO" INTEGER NOT NULL,
  "TITULO" VARCHAR2(30),
  "FECHA_SUBIDA" DATE,
  "USUARIO" INTEGER,
  "VISIBILIDAD" VARCHAR2(20),
  "VALORACION" INTEGER,
  "RUTA" VARCHAR2(40),
  CONSTRAINT "FOTOS_PK" PRIMARY KEY ("IDFOTO"),
  CONSTRAINT "USU_FOTOS_FK" FOREIGN KEY ("USUARIO") REFERENCES USUARIOS("IDUSU"),
  CONSTRAINT "VISIBILIDAD" CHECK (VISIBILIDAD IN ('publica', 'amigos', 'privada')),
  CONSTRAINT "VALORACION" CHECK (VALORACION<=10 AND VALORACION>=0)
);
```

```
CREATE TABLE COMENT_FOTOS (
  "IDCOM" INTEGER NOT NULL,
  "COMENTARIO" VARCHAR2(100),
  "FECHA" DATE,
  "USUARIO" INTEGER,
  "FOTO" INTEGER,
  CONSTRAINT "COMENT_PK" PRIMARY KEY ("IDCOM"),
  CONSTRAINT "USU_COMENT_FK" FOREIGN KEY ("USUARIO") REFERENCES USUARIOS("IDUSU"),
  CONSTRAINT "FOTO_FK" FOREIGN KEY ("FOTO") REFERENCES FOTOS("IDFOTO")
);

DROP SEQUENCE SEQ_USUARIOS;
DROP SEQUENCE SEQ_FOTOS;
DROP SEQUENCE SEQ_COMENTARIOS;

CREATE SEQUENCE SEQ_USUARIOS INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE SEQ_FOTOS INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE SEQ_COMENTARIOS INCREMENT BY 1 START WITH 1;

INSERT INTO USUARIOS VALUES(SEQ_USUARIOS.NEXTVAL,
  'Juan', 'Pérez Rodríguez', TO_DATE('1981/10/5', 'yyyy/mm/dd'),
  'jperez@example.com', 'jperez', 'jperez',
  'Ausente', SYSDATE);
INSERT INTO USUARIOS VALUES(SEQ_USUARIOS.NEXTVAL,
  'Pedro', 'González Rodríguez', TO_DATE('1976/09/12', 'yyyy/mm/dd'),
  'pgonzalez@example.com', 'pgonzalez', 'pgonzalez',
  'Ausente', SYSDATE);
INSERT INTO USUARIOS VALUES(SEQ_USUARIOS.NEXTVAL,
  'Luisa', 'Sancho Martínez', TO_DATE('1985/1/26', 'yyyy/mm/dd'),
  'lsancho@example.com', 'lsancho', 'lsancho',
  'Ausente', SYSDATE);
INSERT INTO USUARIOS VALUES(SEQ_USUARIOS.NEXTVAL,
  'Mario', 'Martín Madroñero', TO_DATE('1988/3/17', 'yyyy/mm/dd'),
  'mmartin@example.com', 'mmartin', 'mmartin',
  'Ausente', SYSDATE);
INSERT INTO USUARIOS VALUES(SEQ_USUARIOS.NEXTVAL,
  'David', 'Soltero Castroviejo', TO_DATE('1982/6/15', 'yyyy/mm/dd'),
  'dsoltero@example.com', 'dsoltero', 'dsoltero',
  'Ausente', SYSDATE);


INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 1', TO_DATE('2009/12/25 12:01:12', 'yyyy/mm/dd hh24:mi:ss'),
  1, 'publica', 5, '/var/www/user/1/foto1.jpg');
INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 2', TO_DATE('2009/12/25 12:02:10', 'yyyy/mm/dd hh24:mi:ss'),
  1, 'publica', 5, '/var/www/user/1/foto2.jpg');
INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 3', TO_DATE('2009/12/25 12:04:02', 'yyyy/mm/dd hh24:mi:ss'),
  1, 'amigos', 8, '/var/www/user/1/foto3.jpg');
INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 4', TO_DATE('2009/12/25 12:05:02', 'yyyy/mm/dd hh24:mi:ss'),
  1, 'amigos', 7, '/var/www/user/1/foto4.jpg');
INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 5', TO_DATE('2009/12/28 15:05:02', 'yyyy/mm/dd hh24:mi:ss'),
  1, 'privada', 9, '/var/www/user/1/foto5.jpg');
```

b) (2,5 puntos) Procedimiento almacenado llamado FOTOS_USUARIO que reciba como parámetro un nombre de usuario y muestre por pantalla los datos personales del mismo, junto con un listado con los datos de las fotos que ha subido (id, título, fecha, visibilidad, valoración y ruta), ordenadas por fecha (primero la más reciente). En caso de error (usuario no existe, no hay fotos de ese usuario, etc...), deberá mostrarse por pantalla un mensaje de advertencia explicando el error. Al finalizar el listado se deberá mostrar la suma de las valoraciones de todas las fotos del usuario.

```
SET SERVEROUTPUT ON SIZE 8000;
CREATE OR REPLACE PROCEDURE FOTOS_USUARIO( param_usuario VARCHAR2 )
```

```
IS
  existe INTEGER;
  numFotos INTEGER;
  detallesUsu USUARIOS%ROWTYPE;
  CURSOR fotosUsu (propietario VARCHAR2) IS
    SELECT *
    FROM USUARIOS U INNER JOIN FOTOS F ON U.idusu = F.usuario
    WHERE
      U.usuario = propietario
    ORDER BY F.fecha_subida DESC;
  acumValoraciones NUMBER(8);
BEGIN
  SELECT COUNT(*) INTO existe
  FROM USUARIOS U
  WHERE
    U.usuario = param_usuario;
  IF existe > 0 THEN
    SELECT * INTO detallesUsu
    FROM USUARIOS U
    WHERE
     U.usuario = param_usuario;

    DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Detalles del usuario ' || param_usuario);
    DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Nombre: ' || detallesUsu.nombre || ' ' || detallesUsu.apes);
    DBMS_OUTPUT.PUT_LINE('Fecha nacimiento: ' || TO_CHAR(detallesUsu.fecha_nac, 'dd/mm/yyyy
hh24:mi:ss'));
    DBMS_OUTPUT.PUT_LINE('email: ' || detallesUsu.email);
    SELECT COUNT(*) INTO numFotos
    FROM USUARIOS U INNER JOIN FOTOS F ON (U.idusu=F.usuario)
    WHERE
      U.usuario = param_usuario;
    IF numFotos > 0 THEN
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      DBMS_OUTPUT.PUT_LINE('Detalles de las ' || numFotos || ' foto(s) del usuario');
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      acumValoraciones := 0;
      FOR foto IN fotosUsu(param_usuario) LOOP
        DBMS_OUTPUT.PUT_LINE('Foto: '|| foto.idfoto);
        DBMS_OUTPUT.PUT_LINE('Fecha: '|| TO_CHAR(foto.fecha_subida, 'yyyy/mm/dd
hh24:mi:ss'));
        DBMS_OUTPUT.PUT_LINE('Visibilidad: '|| foto.visibilidad);
        DBMS_OUTPUT.PUT_LINE('Valoración: '|| foto.valoracion);
        DBMS_OUTPUT.PUT_LINE('Ruta: '|| foto.ruta);
        DBMS_OUTPUT.PUT_LINE('');
        acumValoraciones := acumValoraciones + foto.valoracion;
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      DBMS_OUTPUT.PUT_LINE('Valoraciones acumuladas: '|| acumValoraciones);
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
    ELSE
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      DBMS_OUTPUT.PUT_LINE('El usuario '|| param_usuario || ' no tiene fotos');
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
    END IF;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Usuario ' || param_usuario || ' no existe');
  END IF;
END;
/
BEGIN
  -- Usuario no existe
  FOTOS_USUARIO('falonso');
  -- Usuario sin fotos
  FOTOS_USUARIO('lsancho');
  -- Usuario OK
```

```
  FOTOS_USUARIO('jperez');
END;
/
BEGIN
  -- Elimina el procedimiento de la BBDD
  EXECUTE IMMEDIATE 'DROP PROCEDURE FOTOS_USUARIO';
END;
/
```

c) (1 punto) Dado que la eliminación en cascada (ON DELETE CASCADE) no se ha incluido en el esquema, crea un disparador llamado TR_DEL_COMENT_FOTOS que se encargue de eliminar los comentarios de una foto en el momento que ésta sea eliminada.

```
INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 1', TO_DATE('2009/12/25 12:01:12', 'yyyy/mm/dd hh24:mi:ss'),
  2, 'publica', 5, '/var/www/user/2/foto1.jpg');
INSERT INTO COMENT_FOTOS VALUES(SEQ_COMENTARIOS.NEXTVAL,
  'Qué bonita!', TO_DATE('2009/12/25 16:00:12', 'yyyy/mm/dd hh24:mi:ss'),
  1, SEQ_FOTOS.CURRVAL);
INSERT INTO COMENT_FOTOS VALUES(SEQ_COMENTARIOS.NEXTVAL,
  '¿Qué tipo de cámara has usado?', TO_DATE('2009/12/25 17:02:50', 'yyyy/mm/dd
hh24:mi:ss'),
  3, SEQ_FOTOS.CURRVAL);
INSERT INTO FOTOS VALUES(SEQ_FOTOS.NEXTVAL,
  'Foto 2', TO_DATE('2009/12/25 12:02:10', 'yyyy/mm/dd hh24:mi:ss'),
  2, 'publica', 5, '/var/www/user/2/foto2.jpg');

SET SERVEROUTPUT ON SIZE 8000;
CREATE OR REPLACE TRIGGER TR_DEL_COMENT_FOTOS
AFTER DELETE ON FOTOS
FOR EACH ROW
BEGIN
  DELETE FROM COMENT_FOTOS C
  WHERE
    C.foto = :OLD.idfoto;
  DBMS_OUTPUT.PUT_LINE('Eliminados '|| SQL%ROWCOUNT || ' comentarios de la'
                       || ' fotografia ' || :OLD.idfoto);
END;
/
BEGIN
  DELETE FROM FOTOS F
  WHERE
    F.usuario = 2;
  ROLLBACK;
END;
/
BEGIN
  -- Elimina el trigger de la BBDD
  EXECUTE IMMEDIATE 'DROP TRIGGER TR_DEL_COMENT_FOTOS';
END;
/
```