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

1) (2 puntos) Calcúlese el tamaño de las siguientes operaciones (que pueden devolver tuplas repetidas), en las que intervienen las relaciones r(A,B), s(A,C), t(B,C) y u(C,D), donde los subrayados indican las claves primarias, con tamaños respectivos 100, 200, 300 y 400:
   a) (0,5 puntos) r(A,B) ⋈ s(A,C)

Solución:

Dado que el atributo común es A y es clave en ambas relaciones, se podrán encontrar a lo sumo el número de tuplas de la relación con menos tuplas, es decir, 100

   b) (0,5 puntos) s(A,C) ⋈ t(B,C)

Solución:

Dado que el atributo común es C y es clave en t, a lo sumo todas las tuplas de s podrán encontrar correspondencia con las tuplas de la relación t (asumiendo repetidos en s), es decir, 200.

   c) (0,5 puntos) r(A,B) ⋈ t(B,C)

Solución:

Dado que el atributo común es B y no es clave ni en r ni en t, todas las tuplas de r podrían encontrar correspondencia con tuplas de la relación t, es decir, a lo sumo 100×300=30.000.

   d) (0,5 puntos) r(A,B) ⋈ u(C,D)

Solución:

Dado que no hay atributos comunes, la condición de la reunión es trivialmente cierta, por lo que el tamaño del resultado es exactamente 100×400=40.000.

2) (2,5 puntos) Crear con SQL de Oracle la tabla r(A: Integer, B: Integer, C: Integer, D: Integer) sin ninguna restricción. Para esta tabla se debe cumplir {A} → {B,C} y se pide programar un disparador que compruebe esta restricción de dependencia funcional cuando se inserte o modifique una tupla en r. Si al insertar o modificar en r no se cumple la restricción, se debe generar una excepción y mostrar un mensaje de error (pero permitiendo la acción).

Solución:

La tabla se define con:

```
CREATE TABLE r(A INTEGER, B INTEGER, C INTEGER, D INTEGER);
```

El disparador que maneja las inserciones y modificaciones sobre r:

```
CREATE OR REPLACE TRIGGER ir_r
  BEFORE INSERT OR UPDATE ON r
  FOR EACH ROW
DECLARE
  excepción_df EXCEPTION;
  v_A r.A%TYPE;
  CURSOR c_df IS
    SELECT A
    FROM
     (SELECT DISTINCT A,B,C
      FROM r)
    GROUP BY A
    HAVING COUNT(A)>1;
BEGIN
  OPEN c_df;
  LOOP
    FETCH c_df INTO v_A;
    EXIT WHEN c_df%NOTFOUND;
    RAISE excepción_df;
  END LOOP;
  CLOSE c_df;
EXCEPTION
  WHEN excepción_df THEN
    DBMS_OUTPUT.PUT_LINE('No se cumple la dependencia funcional A -> B,C en
la tupla ('||:NEW.A||','||:NEW.B||','||:NEW.B||') de la tabla r.');
    DBMS_OUTPUT.PUT_LINE('Aún así, se procede a la modificación.');
  CLOSE c_df;
END;
/
```

3) (3 puntos) Considérese un fichero secuencial indexado primario, denso y sin repeticiones, con campos A: CHAR(5), B: CHAR(400), cuya clave es A y codificación ASCII. Se dispone de asignación enlazada, mapas de bits de existencia para los registros y las direcciones de bloques de disco ocupan 4 bytes. El fichero contiene 100.000 registros y el tamaño de bloque es 2.048 bytes. Se pide calcular:
   a) (1,5 puntos) El factor de bloqueo del fichero de datos y del fichero de índice.

Solución:

Fichero de datos:
$(5+400)*N*8+N+4*8 \leq 2048*8$
Primer sumando:
(5+400): Tamaño del registro en bytes
(5+400)*N: Número de bytes debidos a los N registros que caben en el bloque
(5+400)*N*8 = 3240*N: Número de bits debidos a los N registros que caben en el bloque
Segundo sumando:
N: N bits necesarios para el mapa de bits de existencia
Tercer sumando:
4: Tamaño en bytes de la dirección del siguiente bloque en la asignación enlazada
4*8 = 32: Tamaño en bits de la dirección del siguiente bloque en la asignación enlazada
Lado derecho de la inecuación:
2048: Tamaño del bloque en bytes
2048*8 = 16384: Tamaño del bloque en bits
Resolviendo:

3240*N + N + 32 ≤ 16384
N ≤ (16384-32)/(3240+1)
N ≤ 5,048
Por lo tanto, N=5

Fichero de índice:
(5+4)*N*8+N+4*8 ≤ 2048*8
Primer sumando:
(5+4): Tamaño del registro en bytes (campo clave A de 5 bytes y dirección de bloque de 4 bytes)
(5+4)*N: Número de bytes debidos a los N registros que caben en el bloque
(5+4)*N*8 = 72*N: Número de bits debidos a los N registros que caben en el bloque
Segundo sumando:
N: N bits necesarios para el mapa de bits de existencia
Tercer sumando:
4: Tamaño en bytes de la dirección del siguiente bloque en la asignación enlazada
4*8 = 32: Tamaño en bits de la dirección del siguiente bloque en la asignación enlazada
Lado derecho de la inecuación:
2048: Tamaño del bloque en bytes
2048*8 = 16384: Tamaño del bloque en bits
Resolviendo:
72*N + N + 32 ≤ 16384
N ≤ (16384-32)/(72+1)
N ≤ 224. Es una división exacta, por lo que no se desperdicia ningún bit.
Por lo tanto, N=224.

b) (0,5 puntos) El tamaño en bytes del fichero de datos debido sólo a sus campos A y B.

Solución:

Tamaño = (5+400)*100.000 = 40.500.000 bytes.

c) (0,5 puntos) El tamaño real ocupado en disco del fichero de datos, en número de bloques.

Solución:

Número de bloques ocupados = Número de registros / Factor de bloqueo = 100.000 / 5 = 20.000 bloques.

d) (0,5 puntos) El tamaño en bytes del fichero de datos debido a sus campos A y B y la información de control (mapas de bits y dirección enlazada).

Solución:

Como en cada bloque se necesitan 5 bits (mapa de existencia) + 4*8 bits (dirección de enlace) = 37 bits, en todos los bloques se necesitan 20.000*37 bits = 740.000 bits = 92.500 bytes

Tamaño = 40.500.000 + 92.500 = 40.592.500 bytes.

4) (2,5 puntos) Considérese un árbol B (*no un* B+) con *n*=3 (número máximo de hijos por nodos) y valores enteros del 1 al 11 para la clave de búsqueda.
   a) (0,6 puntos) Escribir las restricciones que debe cumplir.

Solución:

El nodo raíz tiene entre 1 y *n* hijos: 1 – 3.

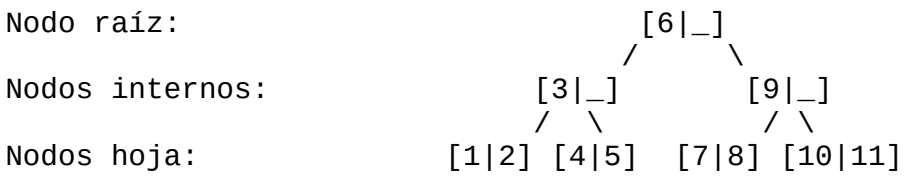
Cada nodo interno (no hoja) tiene entre $\lceil n/2 \rceil$ y *n* hijos: 2 – 3.

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

b) (1,3 puntos) Dibujar el árbol.

Solución:

Como cada nodo puede contener hasta dos valores, se necesitan al menos 11 valores / 2 (valores/nodo) = 6 nodos. Con dos niveles y ocupación máxima es insuficiente (sólo 4 nodos). Con tres niveles y ocupación máxima resultan 3*4+1=13 nodos. Escogemos un árbol con una raíz, 2 nodos internos y 4 hojas, capaz de contener los 11 valores y respetando las restricciones del apartado a).



c) (0,6 puntos) Determinar el número de accesos para alcanzar el fichero de datos bajo una clave de búsqueda si el fichero tuviese 100.000 valores de la clave.

Solución:

Dado un valor tan bajo de *n* (y tan irreal), es de esperar un número muy elevado de niveles:
$\log_3(100.000) = \ln(100.000)/\ln(3) = 10{,}48$. Es decir, 11 niveles del fichero de índices.
Para acceder al fichero de datos: 11 accesos del índice + 1 acceso al fichero de datos = 12 accesos en lectura.