Curso 2008-9
Ingeniería Técnica de Informática
Primer Parcial. 6-Jun-2009.

Nombre:
Grupo:

Se debe entregar esta hoja

1) (3.5 puntos) Se tiene un modelo relacional para un sistema de información con los resultados de las elecciones europeas 2009:
   EuroDiputados(DNI, nombre, apellido, pais, partido, fechaNacimiento*)
   Pais(nombre, superficieKm2, numHabitantes, PIB, añoIngresoUE*)

Realizar las siguientes consultas en SQL:

a) (0,75 puntos) Listar ordenados por edad (los mayores primero) los eurodiputados que han nacido antes de que su país haya ingresado en la Unión Europea y que el PIB de su país sea mayor que la media

```
SELECT DNI,E.NOMBRE,APELLIDO, FECHANACIMIENTO, AÑOINGRESOUE
FROM EURODIPUTADOS E,PAIS P
WHERE PAIS=P.NOMBRE
  AND FECHANACIMIENTO < AÑOINGRESOUE
  AND PIB >  (SELECT AVG(PIB)
                FROM PAIS
            )
ORDER BY FECHANACIMIENTO
```

b) (0,75 puntos) Listar los partidos políticos de tres letras o que empiecen por 'A' representados por más de 5 eurodiputados, mostrando su número de eurodiputados. La salida bebe estar ordenada por orden de representación (primero los que tienen más eurodiputados)

```
SELECT PARTIDO, COUNT(*)
FROM EURODIPUTADOS
WHERE PARTIDO IS LIKE '_ _ _'
 OR PARTIDO IS LIKE 'A%'
GROUP BY PARTIDO
HAVING COUNT(*) > 5
ORDER BY 2 DESC
```

c) (0,75 puntos) Mostrar todos los países ordenados por número de habitantes con el número de eurodiputados que les representan, que debe ser cero para los países que todavía no han ingresado en la UE.

```
SELECT PAIS, COUNT(DISTINCT *),NUMHABITANTES
FROM EURODIPUTADOS RIGHT OUTER JOIN PAIS P
ON PAIS=P.NOMBRE
GROUP BY PAIS, NUMHABITANTES
ORDER BY NUMHABITANTES DESC
```

```
SELECT PAIS, COUNT(*),NUMHABITANTES
FROM EURODIPUTADOS,PAIS P
WHERE PAIS=P.NOMBRE
GROUP BY PAIS, NUMHABITANTES
 UNION
SELECT NOMBRE, 0, NUMHABITANTES
FROM PAIS
WHERE AÑO INGRESOUE IS NULL
ORDER BY NUMHABITANTES DESC
```

d) (0,75 puntos) Mostrar los apellidos de los eurodiputados que tienen otro eurodiputado con el mismo apellido. No debe haber duplicados en la salida.

```
SELECT DISTINCT APELLIDO
FROM EURODIPUTADOS E
WHERE EXISTS (SELECT *
            FROM EURODIPUTADOS
            WHERE APELLIDO = E.APELLIDO
            AND DNI < > E.DNI
             )

SELECT DISTINCT E1.APELLIDO
FROM EURODIPUTADO E1, EURODIPUTADO E2
WHERE E1.APELLIDO = E2.APELLIDO
    AND e1.DNI <> E2.DNI

SELECT APELLIDO
FROM EURODIPUTADO
GROUP BY APELLIDO
HAVING COUNT(*) > 2
```

e) (0,5 puntos) Borrar los eurodiputados cuyo país no aparezca en la tabla de países o no haya ingresado todavía en la UE

```
DELETE FROM EURODIPUTADOS
WHERE PAIS NOT IN (SELECT NOMBRE
                        FROM PAIS)
   OR PAIS IN (SELECT NOMBRE
                FROM PAIS
                WHERE AÑOINGRESOUE IS NULL)
```

2) (3 puntos) Normalización. Dado el conjunto de dependencias funcionales F = {A→BCD, E→ FG, AD→BC} en R(A,B,C,D,E,F,G)

a) (1 punto) Encontrar un conjunto mínimo de dependencias funcionales equivalente

F_{min} = { A→BCD, E→ FG }

Claramente AD→BC se deduce de A → BCD por aumentatividad del atributo D.

También se llega a la misma conclusión descomponiendo F en dependencias elementales ==> AD→BC se decompone en AD→B y AD→C y el atributo D es redundante en ambas, quedándonos A→B y A→C que ya las tenemos y, por tanto se pueden eliminar.

b) (0,5 puntos) Dar las claves candidatas de R

{ A, E } única CC.

Se puede intentar probar a calcular {A, E}+ pues estos dos atributos son los únicos que no están en F_{min} como implicados y, por tanto, necesariamente, deben formar parte de una clave candidata.

{A,E}+ = {A,E,B,C,D,F,G} ≡ R ==> es superclave. Además, es claramente un conjunto mínimo pues no podemos eliminar ni A ni E sin que el subconjunto resultante deje de ser superclave.

.

c) (1 punto) Descomponer R para obtener un conjunto de esquemas en FNBC

R1(A, B, C, D) A→BCD
R21(E, F, G) E→FG

R22(E, A) { }

Aplicando el algoritmo visto en clase. Ver si los implicantes son superclave:

{A}+ = {A,B,C,D}. No es superclave ==> descomponer en

R1(A, B, C, D) A → BCD
R2(E, F, G, A) E → FG y {E}+ = {E,F,G} no es superclave ==> descomponer en

R21(E, F, G) E→FG
R22(E, A) { }

d) (0,5 puntos) ¿La descomposición en FNBC del apartado anterior preserva la propiedad de reunión no aditiva? ¿En dicha descomposición se preservan todas las dependencias funcionales? Si no fuera así descomponer el esquema en 3FN.

Por construcción en el apartado anterior, SÍ que preserva la propiedad de reunión no aditiva.

Claramente también se ve que se preservan todas las dependencias funcionales. Por tanto no será necesario descomponer el esquema anterior en otro esquema de menor nivel como 3FN.

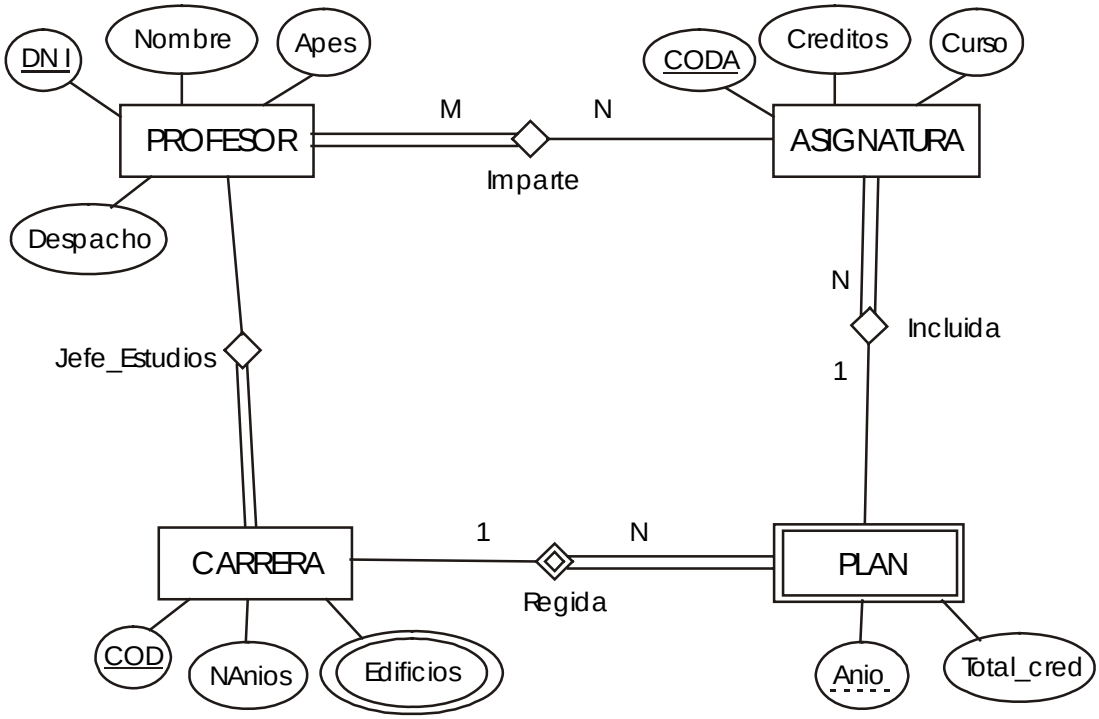
3) (3,5 puntos) Entidad relación y modelo relacional.

a) (2 puntos) Construye el esquema entidad / relación para la base de datos que se describe a continuación, incluyendo atributos clave y restricciones de cardinalidad o participación

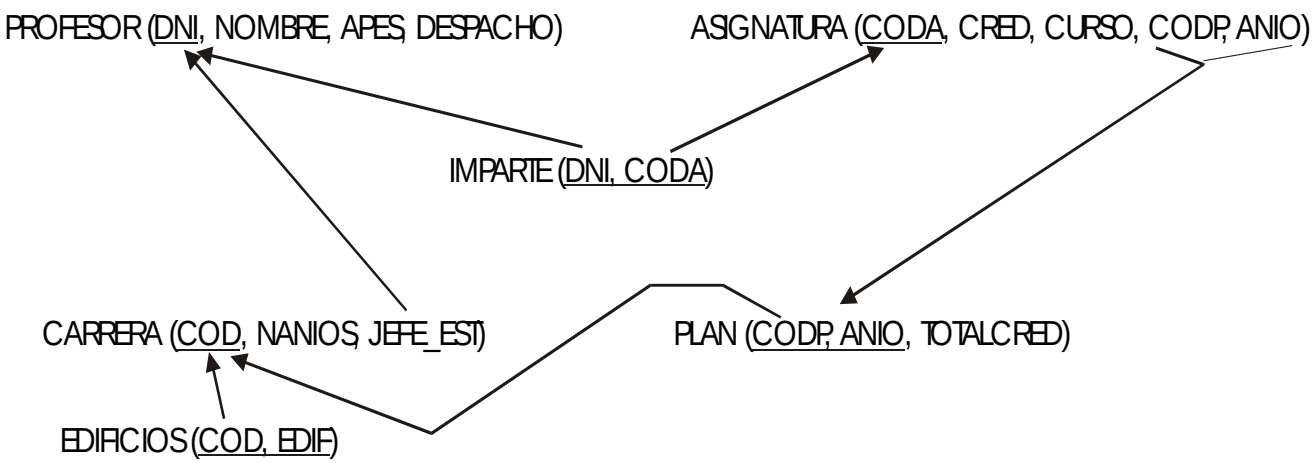
Base de datos sobre profesores y carreras de un centro universitario:

- La base de datos almacenará información sobre las carreras del centro, como por ejemplo, el código de carrera (único), años de duración y edificios donde se imparte cada carrera.
- Se debe almacenar el DNI, nombre, apellidos y número de despacho de cada profesor, pero sólo se quieren almacenar datos sobre profesores que impartan asignaturas.
- Cada carrera tendrá un Jefe de Estudios, que será un profesor, pero un profesor no podrá ser Jefe de Estudios de varias carreras.
- Existirán además varios planes de estudios para cada carrera, de los cuales queremos almacenar su año y el total de créditos. No podrá haber dos planes del mismo año para cada carrera.
- Habrá que almacenar también datos sobre asignaturas: código (único para cada asignatura), número de créditos y curso al que pertenece. Una asignatura sólo existirá si está incluida en un plan de estudios.
- Por último, interesa conocer qué asignaturas imparte cada profesor. Un profesor puede impartir hasta 5 asignaturas y una asignatura puede ser impartida por varios profesores.

b) (1,5 puntos) Transfórmalo después al modelo relacional de manera que obtengas un esquema lo más eficiente posible (sin llegar a normalizar). Realiza las suposiciones que consideres oportunas y razonables acerca de los atributos de cada entidad, anotándolas junto a la solución del ejercicio. Incluye las restricciones de integridad referencia que existan en las tablas resultantes.



Surge una entidad débil, PLAN, porque no hay un atributo que la identifique unívocamente. Depende de la carrera. Además tenemos un atributo multivaluado, Edificios, que podrá tomar varios valores para la misma carrera.

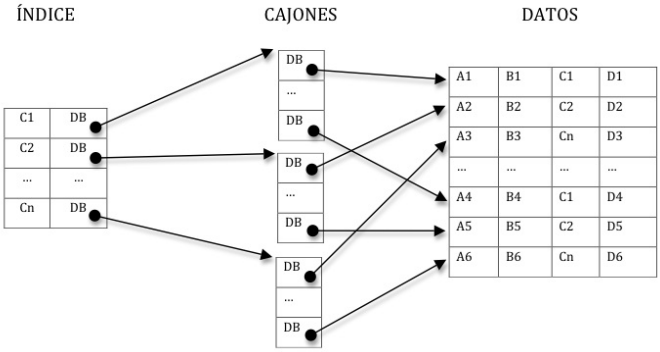


Ficheros y Bases de Datos
Curso 2008-2009
Ingeniería Técnica de Informática, UCM
Segundo parcial. Junio

1) (3 puntos) Se desea indexar la tabla r(A: varchar(10), B: varchar(250), C: varchar(150), D: varchar(490)) de 100.000 filas bajo el atributo C mediante una estructura de índice secundario y asignación enlazada. Las cadenas de caracteres se representan en ASCII (un byte por carácter). Por cada valor distinto de C hay 100 registros de datos con ese valor de C. Los bloques tienen un tamaño de 512 bytes, sus direcciones ocupan 4 bytes y tienen mapas de bits de existencia para los registros. Un cajón se implementa en un bloque. Se pide:
   a) (0,6 puntos) Describir la estructura de los ficheros de índice y de datos, y de los cajones.

Solución (5 minutos):

El conjunto tendrá el siguiente aspecto:



Los VARCHAR son cadenas de longitud variable por lo que necesitan un carácter delimitador adicional.
Fichero de índice:
- Registro:
  - Clave: 150+1=151 bytes.
  - Dirección: 4 bytes.
- Bloque:
  - Mapa de bits de existencia: N bits.
  - N registros.
  - Dirección siguiente bloque: 4 bytes.

Fichero de datos:
- Registro:
  - A: 10+1=11 bytes.
  - B: 250+1=251 bytes.
  - C: 150+1=151 bytes.
  - D: 490+1=491 bytes.
- Bloque:
  - Mapa de bits de existencia: M bits.
  - M registros.
  - Dirección siguiente bloque: 4 bytes.

Cajones:
- Registro:
  - Dirección: 4 bytes.
- Bloque:
  - Mapa de bits de existencia: L bits.
  - L registros.
  - Dirección del siguiente bloque (de desbordamiento): 4 bytes.

b) (0,8 puntos) Calcular el factor de bloqueo de los ficheros de índice y de datos, y de los cajones.

Solución (7 minutos):

Fichero de índice:

$N + N*(151+4)*8 +4*8 \leq 512*8$

$N \leq 3{,}27$, como N debe ser el mayor entero menor o igual que 3,27, entonces N = 3.

Fichero de datos:

El registro ocupa 11+251+151+491 = 904 bytes. Como el bloque tiene 512 bytes, se necesitan dos bloques para alojar un registro (el factor de bloqueo es menor que 1). Si hacemos las cuentas como antes:

$M + M*(904)*8 +4*8 \leq 512*8$

$M \leq 0{,}56$.

Como sólo puede haber parte de un registro en el bloque, no es necesario el mapa de bits. Por lo tanto, en cada bloque podemos alojar 512 (capacidad del bloque) – 4 (tamaño de la dirección del siguiente bloque) = 508 bytes de un registro de datos. Como 904-508=396, esto significa que el primer bloque puede contener los primeros 508 bytes del registro de datos y el segundo bloque los restantes 396 bytes. Esto implica que una operación de lectura requiere dos operaciones de entrada/salida.

Cajones:

$L + L*(4)*8 +4*8 \leq 512*8$

$L \leq 123{,}15$, como L debe ser el mayor entero menor o igual que 123,15, entonces N = 123.

c) (0,6 puntos) Calcular el espacio desperdiciado en los bloques de los ficheros de índice y de datos, y de los cajones.

Solución (5 minutos):

Fichero de índice:

$512*8 - N - N*(151+4)*8 - 4*8 =$ 341 bits = 42 bytes + 5 bits.

Fichero de datos:

512 – 396 – 4 = 112 bytes.

Cajones:

$512*8 - L + L*(4)*8 +4*8 =$ 5 bits

d) (1 punto) Calcular el tamaño en bytes ocupados por los ficheros de índice y de datos, y de los cajones, en los dos casos siguientes: el tamaño neto (sin contabilizar direcciones de enlace y mapas de bits), y el tamaño bruto (calculando el espacio real ocupado). ¿Cuál es el porcentaje de exceso del tamaño bruto con respecto al neto?

Solución (8 minutos):

Fichero de índice:

Tamaño neto:

Como la cardinalidad de la selección de C es 100, esto quiere decir que por cada valor distinto de C se van a encontrar 100 registros de datos con ese valor de C. Por lo tanto, el índice al ser denso contendrá 100.000 / 100 = 1.000 entradas apuntando a los cajones. Por lo tanto, el tamaño neto se calcula como: 100.000 / 100 * 155 = 155.000 bytes.

Tamaño bruto:

Como hay 1.000 entradas y el factor de bloqueo es 3, se necesitan $\lceil 1000/3 \rceil$ = 334 bloques.

Y como cada bloque ocupa 512 bytes, entonces el tamaño bruto es: 334*512 = 171.008 bytes.

Porcentaje de exceso:

$$\frac{Tamaño\_bruto - Tamaño\_neto}{Tamaño\_neto} * 100 = 10{,}3\%$$

Esto significa que se está invirtiendo algo más del 10% del espacio necesario para los datos debido a los mapas de bits, las direcciones de enlace y la fragmentación de los bloques.

Fichero de datos:

Tamaño neto:

Como hay 100.000 entradas (registros) en el fichero de datos y cada registro ocupa 904 bytes, el tamaño neto es: 904*100.000 = 90.400.000 bytes.

Tamaño bruto:

Como cada registro ocupa dos bloques de 512 bytes, son necesarios: 100.000*2*512 = 102.400.000 bytes.

El porcentaje de exceso es:

13,27%

Cajones:

Tamaño neto:

Hay tantas entradas en los bloques como en el fichero de datos, es decir, 100.000, y cada entrada ocupa 4 bytes. Por tanto el tamaño neto es de 400.000 bytes.

Tamaño bruto:

Como hay 100 entradas por cada valor del índice y en cada cajón (un bloque de 512 bytes) caben 123 entradas, bastará con un cajón por cada entrada del índice, es decir, 1.000 cajones. El tamaño bruto será: 1.000*512 = 512.000 bytes.

El porcentaje de exceso es:

28%

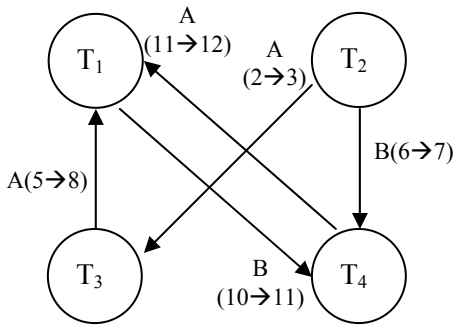
2) (1,5 puntos) Dada la siguiente planificación:

	T_1	T_2	T_3	T_4
1		LOCK(A)		
2		UNLOCK(A)		
3			LOCK(A)	
4		LOCK(B)		
5			UNLOCK(A)	
6		UNLOCK(B)		
7				LOCK(B)
8	LOCK(A)			
9	LOCK(C)			
10				UNLOCK(B)
11	LOCK(B)			
12	UNLOCK(A)			
13				LOCK(A)
14				UNLOCK(A)
15	UNLOCK(B)			
16	UNLOCK(C)			

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

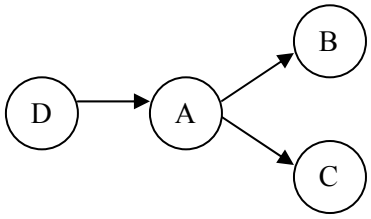
Solución:

Si aplicamos el test de secuencialidad obtenemos el siguiente grafo:



Existe un ciclo entre T1 y T4 por lo que esta planificación no es secuenciable.

b) (0,75 puntos) Dado el siguiente grafo para una base de datos decidir cuáles de las anteriores transacciones siguen el protocolo en árbol.



Solución:

T1 y T3 son válidas siguiendo el protocolo basado en árbol. Sin embargo, T2 y T4 no son válidas. T2 desbloquea A antes de bloquear B por lo que viola la regla de que los sucesivos bloqueos han de producirse sobre elementos cuyo padre ha sido bloqueado por Ti. Lo mismo ocurre con T4, que bloquea antes B que A.

3) (2 puntos) Dada la serie de números: 5, 23, 25, 13, 21, 17, 12, 27, 8, 48, 16, 20
Solución (25 minutos):

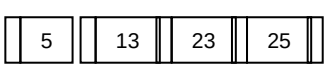
Como se tiene n=5 por nodo, o lo que es lo mismo 5 punteros por nodo, ello implica n-1=4 valores clave por nodo. Dado que todos los valores clave (12 valores) se deben representar en los nodos hoja, se necesitarán al menos [12/4] = 3 nodos hoja.

Las restricciones que debe cumplir el árbol son:

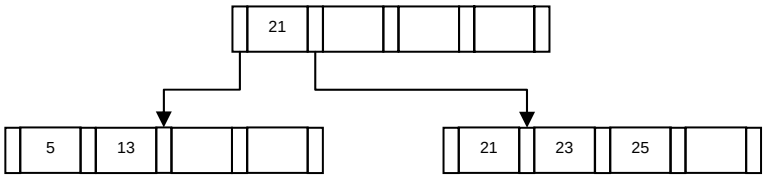
El nodo raíz tiene entre 1 y 5 hijos
Cada nodo interno entre [5/2] = 2 y 4 hijos
Los nodos hoja contienen entre [(5-1)/2]=2 y 4 valores

a) (1,5 puntos) Constrúyase un árbol B+ con n=5 (número de hijos por nodo) insertando uno a uno los números de la serie y en el orden en que aparecen en dicha serie.
Solución (18 minutos):

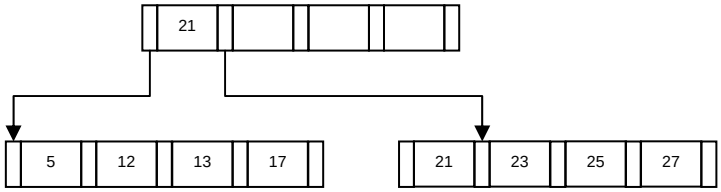
Después de insertar los cuatro primeros



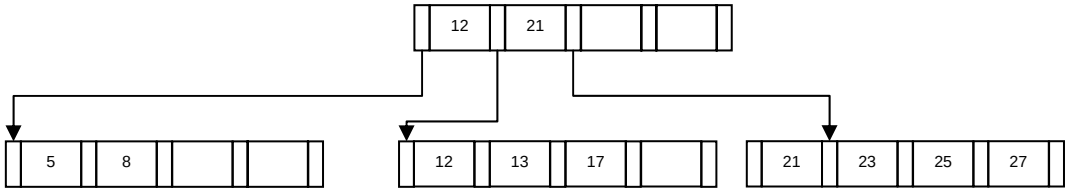
Insertar 21: Cuando necesitamos dividir el nodo dejamos 2 valores a la izquierda y 3 a la derecha. Al final se muestra el resultado dejando 3 valores a la izquierda y 2 a la derecha



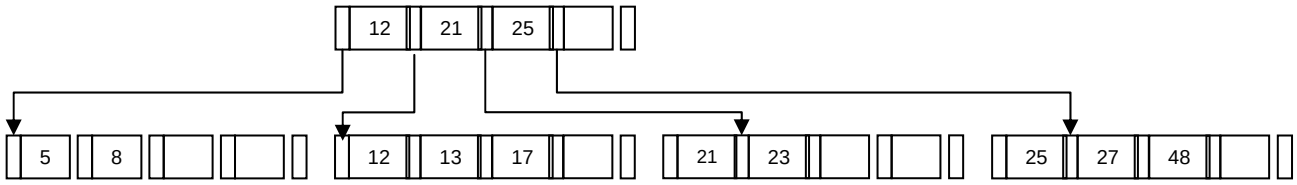
Insertar 17, 12, 27



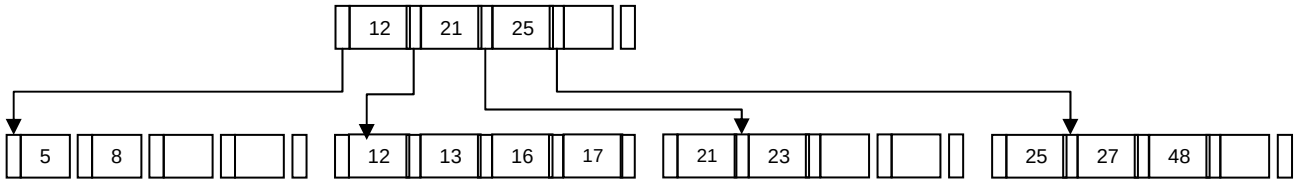
Insertar 8



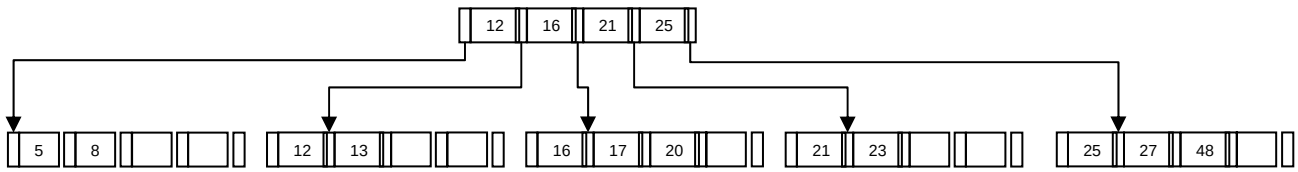
Insertar 48



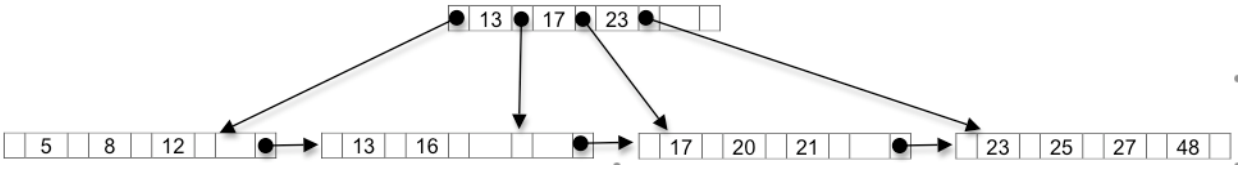
Insertar 16



Insertar 20

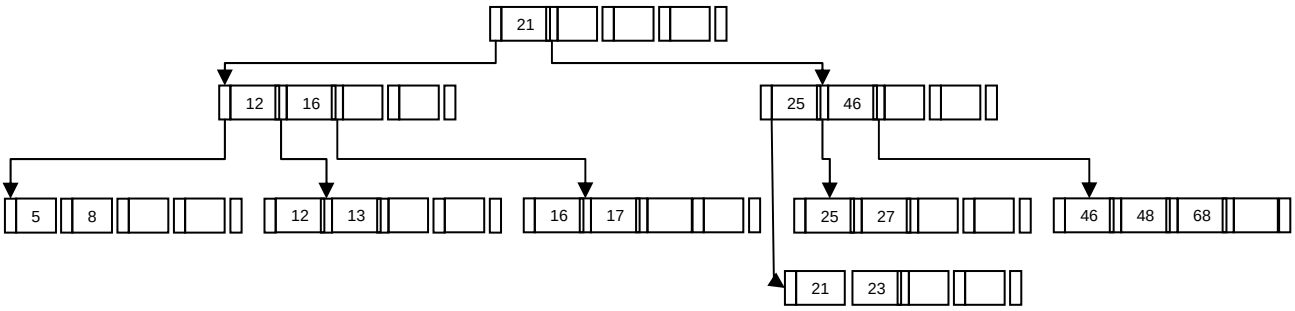


El árbol B+ final si las divisiones se hacen dejando 3 valores a la izquierda y 2 a la derecha será el siguiente:

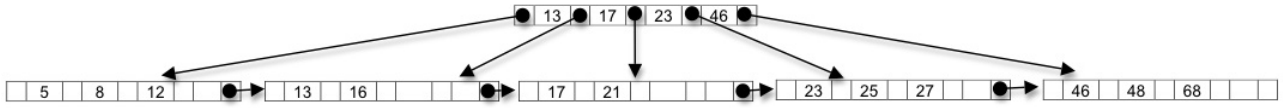


b) (0,5 puntos) Muéstrese el árbol final después de insertar los valores 46 y 68 y borrar el valor 20.
Solución (7 minutos):

Después de insertar el 46, 68 y borrar el 20



Partiendo del segundo árbol, el árbol B+ final si las divisiones se hacen dejando 3 valores a la izquierda y 2 a la derecha será el siguiente:



4) (3,5 puntos) Dadas las tablas PLATOS, PEDIDOS y CONTIENE, similares a la de las prácticas y descritas más adelante, se pide:
   a) (0,75 puntos) Escribir el código SQL para crear las tres tablas teniendo en cuenta lo siguiente:
      - El tipo de datos para las fechas es el tipo DATE de Oracle, para el resto se puede utilizar el tipo de datos que se considere oportuno.
      - Las sentencias DDL deben incluir al menos las sentencias de gestión de integridad referencial. Supóngase que existe una tabla RESTAURANTES cuya clave es "código" para las referencias que lo necesiten (un número de 8 cifras).
      - Incluir las restricciones adicionales que se consideren oportunas.

Solución:

```
CREATE TABLE Platos(restaurante NUMBER(8) REFERENCES Restaurantes("código")
ON DELETE CASCADE, nombre CHAR(20), precio NUMBER(8,2), "categoría" CHAR(10),
"total vendidos" NUMBER(4), PRIMARY KEY(restaurante, nombre));

CREATE TABLE Pedidos("código" NUMBER(8) NOT NULL, estado CHAR(9) DEFAULT
'REST', fecha_hora_pedido DATE, fecha_hora_entrega DATE, "importe total"
NUMBER(8,2), PRIMARY KEY("código"));

CREATE TABLE Contiene(restaurante NUMBER(8), plato CHAR(20), pedido NUMBER(8)
REFERENCES Pedidos("código") ON DELETE CASCADE, unidades NUMBER(4), PRIMARY
KEY(restaurante, plato, pedido), FOREIGN KEY(restaurante, plato) REFERENCES
Platos(restaurante, nombre) ON DELETE CASCADE);
```

b) (1,5 puntos) Codifica un bloque PL/SQL anónimo que actualice el total de platos vendidos en función de los pedidos ya realizados y que incluya un cursor. Se supone que las tablas PEDIDOS y CONTIENE ya existen y contienen datos.

Solución:

```
declare
   cursor cplatos is
     select restaurante, plato, unidades
     from contiene;
   regplatos cplatos%rowtype;
   cuantos integer;
 begin
   -- Inicializamos a 0
   update platos set "total vendidos" = 0;
   for regplatos in cplatos loop
     select "total vendidos" into cuantos
     from platos
     where (platos.restaurante = regplatos.restaurante) and (platos.nombre =
regplatos.plato);
     -- Actualizamos el número de platos vendidos
     cuantos:=cuantos + regplatos.unidades;
     update platos set "total vendidos" = cuantos where restaurante =
regplatos.restaurante and nombre = regplatos.plato;
   end loop;
 end;
 /
 show errors;
```

Otra solución posible realizando agrupaciones:

```
declare
   cursor cplatos is
     select restaurante, plato, sum(unidades) total
     from contiene
     group by restaurante, plato;
   regplatos cplatos%rowtype;
   cuantos integer;
 begin
   for regplatos in cplatos loop
     update platos set "total vendidos" = regplatos.total
     where restaurante = regplatos.restaurante and nombre = regplatos.plato;
   end loop;
 end;
 /
 show errors;
```

c) (1,25 puntos) Escríbase el código de un disparador asociado a la tabla CONTIENE que actualice el total de platos vendidos de PLATOS añadiendo el nuevo número de unidades. La actualización también ha de realizarse cuando se modifique el número de unidades de un plato de un pedido o cuando un plato se elimine de un pedido.

Solución:

```
create or replace trigger actualizar_total_platos
 after delete or insert or update on contiene
 for each row
 declare
   difPlatos integer;
```

```
   -- Procedimiento para realizar las actualizaciones
   procedure actualizarPlatos (rest Platos.restaurante %type, nom
Platos.nombre%type, nPlatos integer) is
   nActual platos."total vendidos"%type;
   begin
     select "total vendidos" into nActual
     from platos
     where nombre = nom and restaurante = rest;

     nActual :=nActual + nplatos;
     update platos set "total vendidos" = nActual where nombre = nom and
restaurante = rest;
   end;
 begin
   if inserting then
     -- Cuando insertamos
     actualizarPlatos(:NEW.restaurante, :NEW.plato, :NEW.unidades);
   elsif updating then
     -- Cuando actualizamos
     difPlatos:= :NEW.unidades - :OLD.unidades;
     actualizarPlatos(:NEW.restaurante, :NEW.plato, difPlatos);
   else
     -- Cuando borramos
     difPlatos:= -(:OLD.unidades);
     actualizarPlatos(:OLD.restaurante, :OLD.plato, difPlatos);
   end if;
 end;
 /
 show errors;
```