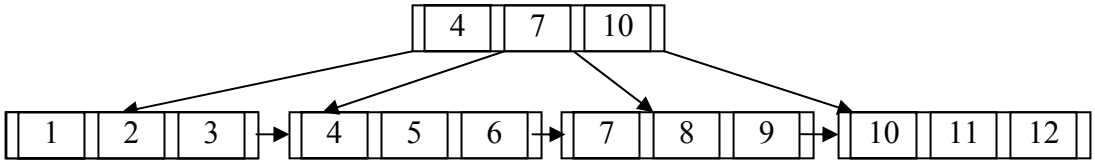
Examen de Ficheros y bases de datos (cód. 520)
Ingeniería Técnica en Informática de Gestión
Convocatoria de junio
II PARCIAL

1) (2 puntos) Dada la serie de números enteros consecutivos del 1 al 12, se pide crear:
   a) (1 punto) Un árbol B+ con *n*=4 (número de hijos por nodo).

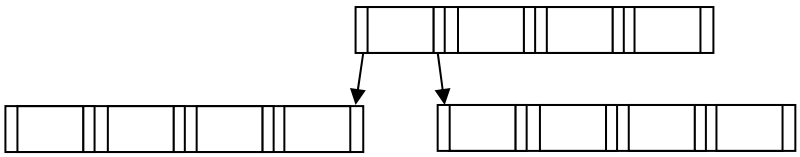
Solución:
Con *n*=4 se pueden alojar *n*-1=3 valores de la clave por nodo. Como todos los valores se deben representar en los nodos hoja, son necesarios $\left\lceil \frac{12}{3} \right\rceil = 4$. Como restricciones que debe cumplir este árbol se tiene:
El nodo raíz tiene entre 1 y 4 hijos.
Cada nodo interno tiene entre $\left\lceil \frac{4}{2} \right\rceil = 2$ hijos y 4.
Los nodos hoja contienen entre $\left\lceil \frac{4-1}{2} \right\rceil = 2$ y 4-1=3 valores.

Una posibilidad para rellenar los valores con las restricciones indicadas es:

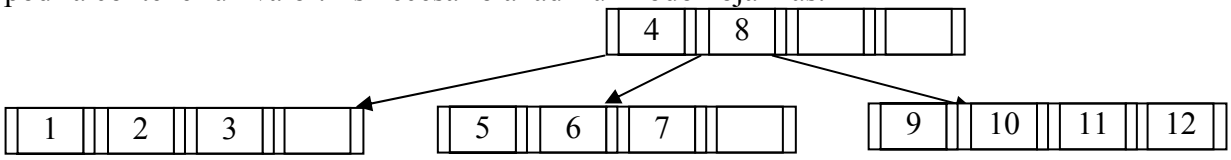


   b) (1 punto) Un árbol B con *n*=5 (supóngase *n*=*m*).

Solución:
Con *n*=5 se necesitan $\left\lceil \frac{12}{4} \right\rceil = 3$ nodos al menos:



Sin embargo, no son suficientes debido a las restricciones estructurales del árbol, ya que el nodo raíz sólo podría contener un valor. Es necesario añadir un nodo hoja más.



Solución (15 minutos):

2) (2,3 puntos) Dadas las siguientes tablas:
   - Pedidos(Código: CHAR(5), Factura: CHAR(10), Precio: NUMBER(6,4)). Significado: se ha pedido un producto (identificado por Código) y que se anota en una factura de código Factura por un precio total de Precio.

- Facturas(Factura: CHAR(10), Importe: NUMBER(8,2)). Significado: la factura de código Factura tiene un importe total de Importe (cada factura puede contener varios pedidos).

a) (0,7 puntos) Escribir las instrucciones SQL para la creación de tablas con restricciones de clave, existencia e integridad referencial donde se estime oportuno.

Solución:

Restricciones de Facturas:
Clave primaria = (Código)
Precio mayor o igual que 0.
Dado que Código es clave primaria, tiene asociada restricción de existencia. NO se debe añadir NOT NULL a este campo.

```
CREATE TABLE Facturas(  Factura CHAR(10) PRIMARY KEY,
                        Importe NUMBER(8,2),
                        CHECK (Importe >= 0));
```

Restricciones de Pedidos:
Integridad referencial con Facturas con el campo factura. Esta restricción se declara inicialmente diferida porque será necesario para resolver el siguiente apartado.
Existencia sobre Precio.
Clave primaria = (Código, Factura)
Precio estrictamente mayor que 0.
Dado que Código y Factura son clave primaria, tienen asociada restricción de existencia. NO se debe añadir NOT NULL a estos campos.

```
CREATE TABLE Pedidos(   Código CHAR(5),
                        Factura CHAR(10) REFERENCES Facturas INITIALLY DEFERRED,
                        Precio NUMBER(8,6) NOT NULL,
                        PRIMARY KEY (Código, Factura),
                        CHECK (Precio > 0));
```

Las tablas se deben crear en este orden para poder declarar la integridad referencial de Pedidos.Factura.

b) (1,6 puntos) Implementar un disparador que calcule automáticamente el contenido de la tabla Facturas cada vez que se añada o borre un pedido en la tabla Pedidos asumiendo que las facturas ya están dadas de alta.

Solución:

Si se inserta una nueva tupla en Pedidos y ya existe la factura correspondiente, simplemente se actualiza la suma en Facturas con todos los pedidos que correspondan a la factura del pedido insertado. Si no existe la factura, entonces se crea.
Si se borra una tupla de Pedidos, es posible que ésta sea la única para su factura. Si es así, se borra la tupla correspondiente de Facturas, en caso contrario, se resta a su importe el precio del pedido borrado.

```
CREATE OR REPLACE TRIGGER ActualizaFactura
  BEFORE INSERT OR DELETE ON Pedidos
  FOR EACH ROW
DECLARE
  v_Cuenta INTEGER;
BEGIN
  IF INSERTING THEN
    SELECT COUNT(*) INTO v_Cuenta FROM Facturas WHERE Factura=:NEW.Factura;
    IF v_Cuenta = 0 THEN
      INSERT INTO Facturas VALUES (:NEW.Factura, 0);
    END IF;
    UPDATE Facturas SET Importe = Importe + :NEW.Precio
```

```
      WHERE Facturas.Factura = :NEW.Factura;
  END IF;
  IF DELETING THEN
    SELECT Importe INTO v_Cuenta FROM Facturas WHERE Factura = :OLD.Factura;
    v_Cuenta := v_Cuenta - :OLD.Precio;
    IF v_Cuenta = 0 THEN
      DELETE FROM Facturas WHERE Factura = :OLD.Factura;
    ELSE
      UPDATE Facturas SET Importe = Importe - :OLD.Precio
        WHERE Facturas.Factura = :OLD.Factura;
    END IF;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error inesperado.');
END;
/
```

Para comprobar este disparador se inserta una nueva tupla en Pedidos, estando ambas tablas vacías.

```
INSERT INTO pedidos VALUES('p1','f1',1);
COMMIT;
SELECT * FROM facturas;
SELECT * FROM pedidos;

1 fila creada.
```

El resultado es:

```
Validación terminada.

FACTURA       IMPORTE
---------- ----------
f1                 1

1 fila seleccionada.

CÓDIG FACTURA        PRECIO
----- ---------- ----------
p1    f1                 1

1 fila seleccionada.
```

Si se añade otro pedido:

```
INSERT INTO Pedidos VALUES('p2','f1',2);
COMMIT;
SELECT * FROM Facturas;
SELECT * FROM Pedidos;

Validación terminada.

FACTURA       IMPORTE
---------- ----------
f1                 3

1 fila seleccionada.

CÓDIG FACTURA        PRECIO
----- ---------- ----------
p1    f1                 1
p2    f1                 2

2 filas seleccionadas.
```

Si se borra este último pedido:

```
DELETE FROM Pedidos WHERE Código='p2';
COMMIT;
SELECT * FROM Facturas;
SELECT * FROM Pedidos;

Validación terminada.

FACTURA       IMPORTE
---------- ----------
f1                  1

1 fila seleccionada.

CÓDIG FACTURA        PRECIO
----- ---------- ----------
p1    f1                 1

1 fila seleccionada.
```

Si se borran todos los pedidos de una factura, ésta también desaparece:

```
DELETE FROM Pedidos;
COMMIT;
SELECT * FROM Facturas;
SELECT * FROM Pedidos;

Validación terminada.

ninguna fila seleccionada

ninguna fila seleccionada
```

Solución (5 minutos):

Restricciones:

- Clave primaria: NIF, CódEstudio.
- Restricciones de existencia: Nombre, Apellidos, CódEstudio, Estudio, Facultad, Dirección.
- Dependencias funcionales (aparte de las derivadas de la clave):
  - {CódEstudio} → {Estudio, Facultad}
  - {Facultad} → {Dirección}
  - {NIF}→ {Nombre, Apellidos}

Implementación de la tabla:

```
CREATE TABLE Alumnos(       NIF CHAR(10) PRIMARY KEY,
                            Nombre CHAR(50) NOT NULL,
                            Apellidos CHAR(100) NOT NULL,
                            CódEstudio CHAR(5) NOT NULL,
                            Estudio CHAR(50) NOT NULL,
                            Facultad CHAR(50) NOT NULL,
                            Dirección CHAR(100) NOT NULL,
                            PRIMARY KEY (NIF,CódEstudio));
```

Las dependencias funcionales no se puede implementar directamente en SQL.

Solución (15 minutos):

Crearemos un procedimiento para detectar la dependencia funcional {Facultad} → {Dirección}

3) (4,5 puntos) Considérese un fichero secuencial secundario e indexado de dos niveles, y organizado sobre una clave con repeticiones. El primer nivel tiene tantas entradas como letras del alfabeto (supongamos 28) y cada entrada es el primer valor de la clave que existe en el fichero de datos que empieza por cada una de las letras del alfabeto (por ejemplo: arturo, beatriz, carlos, ...). Se supone que al menos hay un registro en el fichero de datos por cada una de estas entradas. El segundo nivel es denso. En media se tienen 100 entradas en el segundo nivel por cada letra del alfabeto y 500 entradas del fichero de datos por cada valor de la clave. Se dispone de asignación enlazada, mapas de bits de existencia para los bloques, un tamaño de bloque de 1.024 bytes y direcciones de bloque de 4 bytes. El registro de datos tiene tres campos: A (campo clave de 15 caracteres), B (20 caracteres) y C (20 caracteres) con codificación ASCII. Se pide:
   a) (0,5 puntos) ¿Cuántos registros tienen cada uno de los dos niveles del índice y el fichero de datos?

Solución:

Primer nivel: 28 registros (como indica el enunciado).
Segundo nivel: 28 registros × 100 (100 registros en media por cada letra del alfabeto) = 2.800 valores de la clave diferentes.
Fichero de datos: 2.800 × 500 (por cada valor de la clave diferente que aparece en el segundo nivel, hay 500 registros en media en el fichero de datos) = 1.400.000 registros en el fichero de datos.

   b) (1,6 puntos) Calcular el factor de bloqueo en los dos niveles del fichero de índices y en el fichero de datos. Asumir que cada nivel del índice está implementado con un fichero independiente.

Fichero de índice:
La estructura de los dos niveles del índice es igual: un campo para la clave (A, de 15 bytes), otro para la dirección de bloque correspondiente a la clave, mapa de bits de existencia y dirección de bloque por la asignación enlazada. Por tanto, el factor de bloqueo N se calcula como:
$N \times (15+4) \times 8 + N + 4 \times 8 \leq 1.024 \times 8$
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.
$N \leq 53,33$
Por lo tanto, escogemos el entero N = 53.

Fichero de datos:
La estructura del fichero de datos es: campos de datos (A, de 15 bytes, B y C de 20), mapa de bits de existencia y dirección de bloque por la asignación enlazada. Por tanto, el factor de bloqueo N se calcula como:
$N \times (15+20+20) \times 8 + N + 4 \times 8 \leq 1.024 \times 8$
$N \leq 18,50$
Por lo tanto, escogemos el entero N = 18.

   c) (0,8 puntos) En el caso medio, calcular el espacio desperdiciado en el bloque de un cajón.

La estructura del cajón es: direcciones de bloque al fichero de datos, mapa de bits de existencia y dirección de bloque por la asignación enlazada. Por tanto, el factor de bloqueo N se calcula como:
$N \times (4) \times 8 + N + 4 \times 8 \leq 1.024 \times 8$
$N \leq 247,27$
Por lo tanto, escogemos el entero N = 247.

Un bloque es insuficiente para almacenar las direcciones de los 500 valores repetidos. Serán necesarios 3 bloques (247+247= 494, faltan 6 valores que irán a un tercer bloque). El espacio desperdiciado del tercer bloque será:
1.024 × 8 (espacio total) - 6 × 4 × 8 (registros ocupados) - 247 (mapa de bits) - 4 × 8 (dirección por la asignación enlazada) = 7.721 bits = 965,125 bytes = 965 bytes y 1 bit.

d) (1,6 puntos) ¿Cuál es el tiempo medio necesario para la lectura del primer registro según el valor de la clave? Considérese un disco de 7.200 rpm con un tiempo de búsqueda de 10 ms y 128 sectores por pista.

En primer lugar se calcularán cuántos bloques es necesario leer, después el tiempo necesario para la lectura de un bloque y finalmente el tiempo total.

1- Cálculo del número de bloques a leer.
El número de bloques en media que es necesario leer para la lectura de un registro del fichero de datos a partir del valor de la clave es:
Primer nivel del índice. Dado que sólo hay 28 valores en el primer nivel y el factor de bloqueo es 53, sólo hay un bloque en el primer nivel. Por lo tanto, una lectura de bloque en el primer nivel.
Segundo nivel del índice. Como en el segundo nivel hay 100 registros en media por cada valor de la clave y el factor de bloqueo también es 53, se necesitan leer $\left\lceil \frac{1}{2} \cdot \frac{100}{53} \right\rceil = 1$ bloque. Por tanto, 1 lectura de bloque en el segundo nivel.
Cajón. Si sólo se quiere leer el primero, sólo una lectura de bloque. Si se quiere leer alguno en concreto que cumpla otra condición, en media serían dos lecturas de bloque.
Fichero de datos. Sólo una lectura porque es un índice secundario y localizamos exactamente el bloque en que se encuentra a partir del cajón.
Por lo tanto: 1 (primer nivel) + 1 (segundo nivel) + 2 (peor caso cajón) + 1 (datos) = 6 lecturas de bloque.

2- Cálculo del tiempo de lectura de bloque.
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=10ms
Tiempo de latencia=1/2*Inversa de la frecuencia = $\frac{1}{2} \frac{1}{7.200 \frac{1}{\min} \frac{1\min}{60.000 ms}} = 4{,}17 ms$
Tiempo de transmisión= $\frac{1}{128}$ *Tiempo de rotación(el doble del tiempo de latencia)=0,07ms
Tiempo medio operación de lectura de un bloque = 10+4,17+0,07=14,24 ms

3- Tiempo total 14,24 × 6 = 85,44 ms.

Solución (10 minutos):

Para calcular el grado máximo de salida hay que analizar cuántos valores de la clave caben en un bloque. Estudiamos su estructura:
N punteros a bloques (4 bytes), N-1 valores de la clave (128 bytes) y N-1 bits para el mapa de bits de existencia:
N*4*8+(N-1)*128*8+N-1 ≤ 1.024*8
N*32+N*1.024-1024+N-1 ≤ 8.192

N*(32+1.024+1) ≤ 8.192+1.024+1
N ≤ $\lfloor 8,7 \rfloor$ = 8, que es el grado de salida máximo
El espacio desperdiciado es:
8.192 - (8*32+8*1.024-1024+8-1) = 761 bits, es decir, 95 bytes y 1 bit.
La estructura queda:

|Mapa de bits | Puntero 0 | Valor 1 | Puntero 1 | Valor 2 | ... | Valor 7 | Puntero 7|

Solución (10 minutos):

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=10ms
Tiempo de latencia=1/2*Inversa de la frecuencia = $\frac{1}{2}\frac{1}{7.200\frac{1}{\min}\frac{1\min}{60.000ms}} = 4{,}17ms$
Tiempo de transmisión=$\frac{1}{128}$*Tiempo de rotación(el doble del tiempo de latencia)=0,07ms
Tiempo medio operación E/S=10+4,17+0,07=14,24 ms

Ahora hay que calcular cuántos niveles hay que atravesar hasta encontrar el registro en el nodo hoja:
$\lceil \log_8 10^6 \rceil = 7$

El tiempo de acceso será la lectura de estos 7 bloques más la del bloque en que se encuentra el dato (en el fichero de datos):
Tiempo de acceso con fichero de índices = 8*14,24=114ms

Sin el fichero de índices habría que recorrer 500.000 registros en media, es decir, 500.000 bloques:
Tiempo de acceso sin fichero de índices = 500.000 * 14,24 = 7.120 s = 119 minutos = ¡casi dos horas!

4) (1,2 puntos) Dadas las siguientes transacciones, determinar el valor semántico de la planificación secuencial en la que T1 precede a T2.

T1		**T2**	
LOCK A		LOCK B	
LOCK B		LOCK C	
UNLOCK A	$f_1(A,B)$	UNLOCK B	$f_3(B,C)$
UNLOCK B	$f_2(A,B)$	LOCK A	
		UNLOCK C	$f_4(A,B,C)$
		UNLOCK A	$f_5(A,B,C)$

Solución:

T1	**T2**	**A**	**B**	**C**
LOCK A		A_0	B_0	C_0
LOCK B		A_0	B_0	C_0
UNLOCK A		$f_1(A_0,B_0)$	B_0	C_0
UNLOCK B		$f_1(A_0,B_0)$	$f_2(A_0,B_0)$	C_0

	LOCK B	$f_1(A_0, B_0)$	$f_2(A_0, B_0)$	C_0
	LOCK C	$f_1(A_0, B_0)$	$f_2(A_0, B_0)$	C_0
	UNLOCK B	$f_1(A_0, B_0)$	$f_3(f_2(A_0, B_0), C_0)$	C_0
	LOCK A	$f_1(A_0, B_0)$	$f_3(f_2(A_0, B_0), C_0)$	C_0
	UNLOCK C	$f_1(A_0, B_0)$	$f_3(f_2(A_0, B_0), C_0)$	$f_4(f_1(A_0, B_0), f_2(A_0, B_0), C_0)$
	UNLOCK A	$f_5(f_1(A_0, B_0), f_2(A_0, B_0), C_0)$	$f_3(f_2(A_0, B_0), C_0)$	$f_4(f_1(A_0, B_0), f_2(A_0, B_0), C_0)$