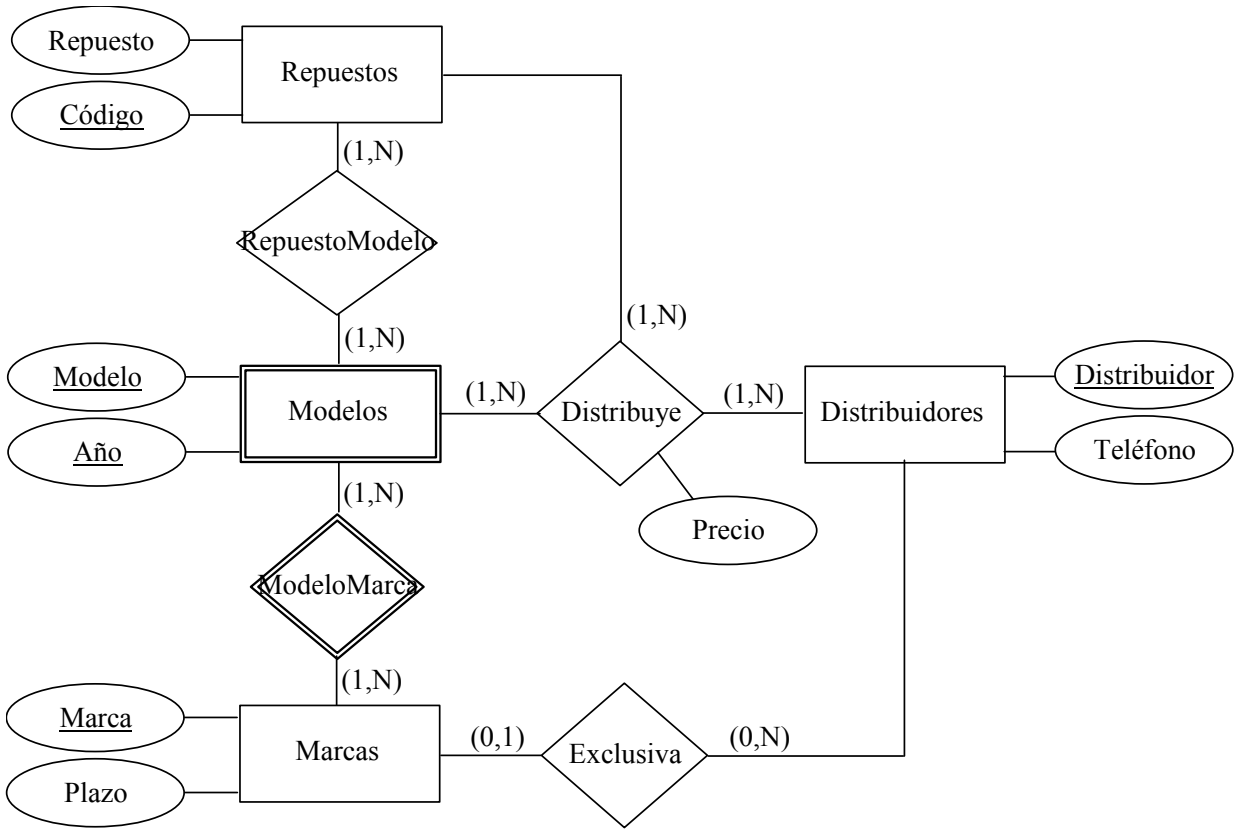
Examen de Ficheros y bases de datos (cód. 520)
Ingeniería Técnica en Informática de Gestión
Convocatoria de septiembre
I PARCIAL

1) (2,2 puntos) Un taller de reparaciones necesita almacenar información sobre los repuestos. Se debe conocer el nombre del repuesto (ej: volante) y su código (ej: VOL) y el modelo de coche en el que se puede montar. Del modelo se debe conocer su nombre y año de entrada en el mercado, así como su marca. De la marca se debe conocer su nombre y el plazo de tiempo sobre el que aseguran tener repuestos. Hay una serie de distribuidores (de los que es necesario conocer su nombre y teléfono) que son los que proporcionan repuestos para modelos en concreto a un precio determinado. Además, algunos de estos distribuidores trabajan en exclusiva para algunas marcas. Considérese que los nombres de modelos pueden repetirse para diferentes marcas o años. Se pide diseñar el diagrama entidad-relación con el mínimo número de atributos necesarios y sin introducir identificadores artificiales. Subrayar las clave primarias y explicar las restricciones de participación.

Solución (15 minutos):

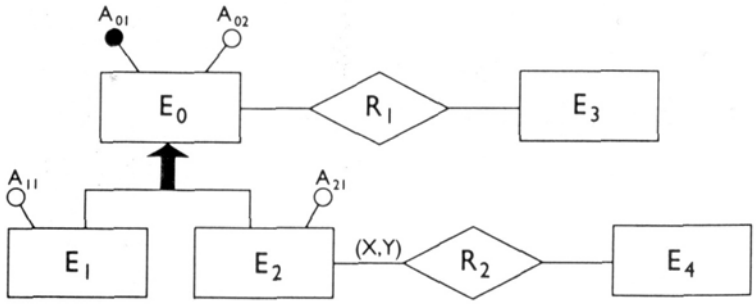


Restricciones de participación:

- Repuestos – (1,N) – RepuestoModelo: Cada repuesto debe corresponder al menos a un modelo de coche. El mismo repuesto puede valer para varios modelos.
- RepuestoModelo – (1,N) – Modelos: Debe haber al menos un repuesto para cada modelo.
- Modelos – (1,N) – ModeloMarca: Cada modelo debe pertenecer a una marca. Puede darse el caso que en un mismo año se produzca un modelo con el mismo nombre de marcas diferentes.
- ModeloMarca – (1,N) – Marcas: De cada marca debe haber uno o más modelos.
- Marcas – (0,1) – Exclusiva: No todas las marcas tienen distribuidores en exclusiva, pero si lo tienen, sólo puede ser uno.
- Exclusiva – (0,N) – Distribuidores: Un distribuidor puede suministrar o no en exclusiva repuestos para una o varias marcas.
- Repuestos – (1,N) – Distribuye: Un repuesto lo puede suministrar uno o más distribuidores para los modelos.
- Modelos – (1,N) – Distribuye: Deben suministrarse repuestos para cada modelo.

- Distribuidores – (1;N) – Distribuye: Cada distribuidor puede suministrar uno o más repuestos para los modelos.

2) (2 puntos) Explicar las diferentes alternativas de traducción del siguiente esquema entidad-relación a esquemas relacionales:

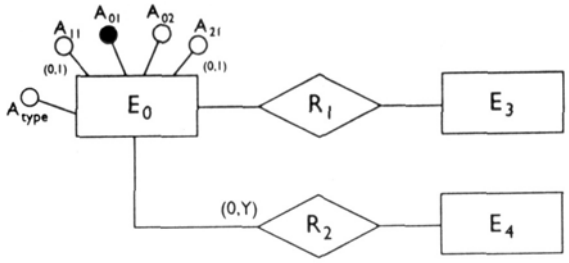


Solución (15 minutos):

Principales alternativas para transformar generalizaciones:

1. Plegar las entidades hijo en la entidad padre.

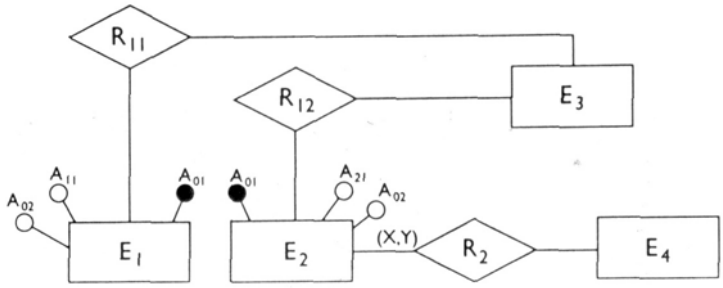
   Adecuada cuando las operaciones implican a los atributos de E0, E1 y E2 más o menos de la misma forma. Aunque mayor espacio de almacenamiento, requiere menos accesos.



   La traducción al esquema relacional:
   E0(A01, A02, A11, A21, Atype)
   E3(A31) Atributo clave no mostrado en la figura
   E4(A41) Atributo clave no mostrado en la figura
   R1(A01, A31)
   R2(A01, A41)

2. Plegar las entidades padre en las entidades hijo.

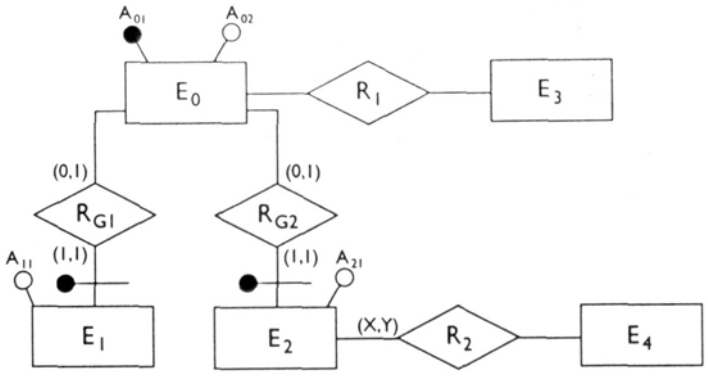
   Sólo es posible si la generalización es total. Si no, las E0 que no son ni de E1 ni de E2 no se representarían. Adecuada cuando para operaciones que se refieren sólo a E1 o E2, se ahorra almacenamiento con respecto a la alternativa 1 (no hay atributos NULL) y hay menos accesos que en la alternativa 3 (no es necesario acceder a E0 para obtener atributos de E1 y E2).



   La traducción al esquema relacional:
   E1(A01, A02, A11)
   E2(A01, A21, A02)
   E3(A31) Atributo clave no mostrado en la figura
   E4(A41) Atributo clave no mostrado en la figura
   R11(A01, A31)
   R12(A01, A31)
   R2(A01, A41)

3. Sustituir la generalización por relaciones.

Adecuada cuando la generalización no es total y las operaciones se refieren a atributos de E1(E2) o de E0 (se distinguen los padres de los hijos). Se ahorra espacio con respecto a la alternativa 1 (no hay atributos NULL), pero hay más accesos para mantener la consistencia.



La traducción al esquema relacional:
E0(A01, A02)
E1(A01, A11)
E2(A01, A21)
E3(A31) Atributo clave no mostrado en la figura
E4(A41) Atributo clave no mostrado en la figura
R1(A01, A31)
RG1(A01)
RG2(A01)
R2(A01, A41)

Parece que la alternativa 3 nunca se usaría. Sin embargo, produce entidades con menos atributos, y los detalles físicos pueden hacerla más efectiva.

3) (3,6 puntos) Dada la siguiente relación *Notas*:

Alumno	*Nota*	*Curso*
1	SS	2004
2	AP	2005
3	AP	2005
4	AP	2004
5	NT	2005
6	MH	2002
7	NT	2004
8	AP	2004
9	SB	2005
10	AP	2004

Se pide:

a) (0,7 puntos) Determinar con SQL el recuento de cada una de las notas del curso 2004 siempre que este recuento sea al menos tres. No se pueden usar instrucciones anidadas.

Solución (5 minutos):

```
SELECT Nota, COUNT(*)
FROM Notas
WHERE Curso='2004'
GROUP BY Nota
HAVING COUNT(*) >= 3;
```

b) (0,7 puntos) Mostrar paso a paso cómo se evaluaría la consulta hasta entregar el resultado final. Indicar las relaciones que se van construyendo en cada paso.

Solución (5 minutos):

Primer paso: Filtro WHERE Curso=2004

Alumno	*Nota*	*Curso*
1	SS	2004
4	AP	2004

7	NT	2004
8	AP	2004
10	AP	2004

Segundo paso: Agrupación GROUP BY Nota

Nota	*Alumno*	*Curso*
SS	1	2004
AP	4	2004
	8	2004
	10	2004
NT	7	2004

Tercer paso: Filtro agrupación HAVING COUNT(*) >=3

Nota	*Alumno*	*Curso*
AP	4	2004
	8	2004
	10	2004

Cuarto paso: Proyección SELECT Nota, COUNT(*)

Nota	*COUNT(*)*
AP	3

c) (0,8 puntos) Generar un listado de notas con SQL que muestre: curso y nota media de todos los alumnos de cada curso. La nota media se calcula a partir del valor numérico de cada nota que se encuentra en la tabla Valores(Nota,Valor)

Solución (5 minutos):

```
SELECT Curso, AVG(Valor) AS Media
FROM Notas, Valores
WHERE Notas.Nota = Valores.Nota
GROUP BY Curso;
```

d) (1,4 puntos) Determinar con SQL, CRD y CRT las notas que aparecen en Valores pero que no aparecen en Notas.

Solución (10 minutos):

SQL:

```
SELECT Nota
FROM Valores
WHERE NOT EXISTS (SELECT * FROM Notas WHERE Notas.Nota = Valores.Nota);
```

CRD:

$\{<n> \mid \exists v(<n,v> \in Valores) \wedge \neg \exists c,a(<c,a,n> \in Notas)\}$

CRT:

$\{t \mid v \in Valores \wedge t[Nota]=v[Nota] \wedge (\neg \exists n \in Notas\ v.Nota = n.Nota)\}$

4) (2,2 puntos) Sea R(A,B,C,D) un esquema de relación y $F=\{A \rightarrow B, BD \rightarrow C, C \rightarrow A\}$ el conjunto de dependencias funcionales asociado. Calcúlese el conjunto F+, las claves candidatas de R y las superclaves.

Solución (15 minutos):

Se calcula el cierre de los 15 subconjuntos de atributos que se pueden formar:
B, D, C, A,
BD, BC, AB, DC, AD, AC,
BCD, ABD, ACD, ABC,
ABCD

Los de un elemento:
B+ = B, D+ = D, C+ = ABC y A+ = AB.
Por tanto, la única dependencia nueva que podríamos añadir con un único atributo en el lado izquierdo es:
C→B.

Los de dos elementos:
BD+ = ABCD, de donde obtenemos la dependencia BD→A.
BC+ = ABC, de donde obtenemos BC→A.
AB+ = AB.
CD+ = ABCD, de donde obtenemos CD→B y CD→A.
AD+ = ABCD, de donde obtenemos AD→B y AD→C.
AC+ = ABC, de donde obtenemos AC→B.

Los de tres elementos:

ABC+ = ABC
BCD+ = ABCD
ABD+ = ABCD
ACD+ = ABCD

Obtenemos BCD→A, ABD→C, y ACD→B.

ABCD+ = ABCD

Las 11 nuevas dependencias son:
{C→B, BD→A, BC→A, CD→B, CD→A, AD→B, AD→C, AC→B, BCD→A, ABD→C, y ACD→B}

Habría que unir el conjunto anterior con F={BD→C, C→A D A→B}
y con el conjunto de todas las dependencias triviales para obtener F+.

¿Cuáles son las claves candidatas de R?

Analizando los cierres que acabamos de obtener, podemos ver que BD, CD y AD son claves porque determinan funcionalmente a todos los atributos de la relación ({A,B,C,D}).
Todos los demás, o bien su cierre no es ABCD o bien son un superconjunto de los anteriores (y por tanto no son claves sino superclaves).

¿Cuáles son superclaves de R pero no son claves candidatas?

Las superclaves son todos los conjuntos cuyo cierre es ABCD y que son superconjuntos de las claves candidatas:
BCD, ABD, ACD y ABCD

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

1) (2,3 puntos) Dada la tabla Presupuesto(Concepto, Gasto, Fecha) que almacena gastos en euros por concepto y fecha (de tipo DATE), se pide programar un disparador que actualice la información de resumen que se encuentra en la tabla Estadísticas(Año, TotalEuros, NúmeroGastos), que almacena para cada año el total en euros de gastos y el número de estos gastos.

Solución (15 minutos):

Creamos las tablas:

DROP TABLE Presupuesto;

```
CREATE TABLE Presupuesto(Concepto CHAR(20), Gasto NUMBER(6,2) NOT NULL, Fecha DATE
NOT NULL), PRIMARY KEY (Concepto, Gasto, Fecha));
CREATE TABLE Estadisticas(Anyo CHAR(4) PRIMARY KEY, TotalEuros NUMBER(10,2),
NumeroGastos INTEGER);
```

El algoritmo que debe implementar el disparador se puede resumir en:

- Si se inserta una tupla es necesario borrar la tupla correspondiente de Estadisticas, si la hubiera, y añadirla de nuevo actualizada.
- Si se borra una tupla, hay que borrar la tupla de Estadisticas correspondiente y añadirla de nuevo actualizada si quedan otras tuplas del mismo año en Presupuesto.
- Si se modifica la fecha de una tupla se procede como un borrado seguido de una inserción.
- Si se modifica una tupla, pero no su fecha, se procede como una inserción.

```
CREATE OR REPLACE TRIGGER estadistica
  AFTER INSERT OR DELETE OR UPDATE ON Presupuesto
  FOR EACH ROW
DECLARE
  v_Anyo CHAR(4);
  v_TotalEuros Estadisticas.TotalEuros%TYPE;
  v_NumeroGastos INTEGER;
BEGIN
  IF INSERTING THEN
    v_Anyo := TO_CHAR(:NEW.Fecha,'YYYY');
    borrar_estadistica(v_Anyo);
    insertar_estadistica(v_Anyo);
  END IF;
  IF DELETING THEN
    v_Anyo := TO_CHAR(:OLD.Fecha,'YYYY');
    borrar_estadistica(v_Anyo);
    IF EXISTS (SELECT * FROM Presupuesto WHERE TO_CHAR(Fecha,'YYYY')=v_Anyo) THEN
      insertar_estadistica(v_Anyo);
    END IF;
  END IF;
  IF UPDATING THEN
    IF :OLD.Fecha <> :NEW.Fecha THEN
      v_Anyo := TO_CHAR(:OLD.Fecha,'YYYY');
      borrar_estadistica(v_Anyo);
      IF EXISTS (SELECT * FROM Presupuesto WHERE TO_CHAR(Fecha,'YYYY')=v_Anyo) THEN
        insertar_estadistica(v_Anyo);
      END IF;
      v_Anyo := TO_CHAR(:NEW.Fecha,'YYYY');
      borrar_estadistica(v_Anyo);
      insertar_estadistica(v_Anyo);
    ELSE
      v_Anyo := TO_CHAR(:NEW.Fecha,'YYYY');
      borrar_estadistica(v_Anyo);
```

```
      insertar_estadistica(v_Anyo);
    END IF;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error inesperado.');
END;
/

CREATE OR REPLACE Procedure borrar_estadistica(v_Anyo CHAR(4)) AS
BEGIN
    DELETE FROM Estadisticas WHERE Anyo = v_Anyo;
END;

CREATE OR REPLACE Procedure insertar_estadistica(v_Anyo CHAR(4)) AS
BEGIN
    INSERT INTO Estadisticas VALUES
     (v_Anyo,
      SELECT SUM(Gasto) FROM Presupuesto WHERE TO_CHAR(Fecha,'YYYY')=v_Anyo,
      SELECT COUNT(*) FROM Presupuesto WHERE TO_CHAR(Fecha,'YYYY')=v_Anyo);
END;
```

Partiendo de las tablas Presupuesto y Estadisticas vacías:

1. Inserción de una primera fila (no se borra de Estadísticas):
   `INSERT INTO Presupuesto VALUES('Boligrafos',10.0,'10/10/2006');`
   Consecuencia:

2. Inserción de otra fila para el mismo año (se borra de Estadísticas):
   `INSERT INTO Presupuesto VALUES('Lapices',10.0,'10/10/2006');`
   Consecuencia:

3. Borrado de una fila (se borra y añade a Estadísticas):
   `DELETE FROM Presupuesto WHERE Concepto='Boligrafos';`
   Consecuencia:

4. Borrado de una fila (se borra pero no se añade a Estadísticas):
   `DELETE FROM Presupuesto WHERE Concepto='Lapices';`
   Consecuencia:

5. Inserción de una primera fila (como en el paso 1):
   `INSERT INTO Presupuesto VALUES('Boligrafos',10.0,'10/10/2006');`

6. Modificación de la fila para otro año (se borra y añade en Estadísticas):
   `UPDATE Presupuesto SET Fecha='10/10/2007' WHERE Concepto = 'Boligrafos';`
   Consecuencia:

7. Modificación de la fila para el mismo año (se borra y añade en Estadísticas):
   `UPDATE Presupuesto SET Concepto='Lapices' WHERE Concepto = 'Boligrafos';`
   Consecuencia:

2) (3,7 puntos) Se requiere implementar en una única tabla información sobre:
   - Alumnos matriculados (NIF:CHAR(10), Nombre:CHAR(50), Apellidos:CHAR(100)). Un alumno debe ser considerado como tal si se ha matriculado al menos en un estudio. Este estudio debe estar ofertado por alguna facultad.
   - Estudios en que se ha matriculado cada alumno (CódEstudio:CHAR(5), Estudio:CHAR(50)). Cada alumno puede estar matriculado en uno o más estudios.
   - Facultad en la que se imparten los estudios (Facultad:CHAR(50), Dirección:CHAR(100)). Cada estudio sólo se imparte en una facultad.

a) (0,7 puntos) Diseñar e implementar la tabla con SQL especificando todas las restricciones de integridad identificadas.

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.

b) (1 punto) Indicar dónde se produce redundancia y calcular el espacio desperdiciado (en número de caracteres) para un volumen de la tabla de 10.000 filas, considerando que se ofertan 50 estudios diferentes por 10 facultades diferentes (cada facultad oferta cinco estudios) y hay matrícula en todos los estudios.

Solución (10 minutos):

Se produce redundancia en los atributos que aparecen como consecuentes en las dependencias funcionales que no son superclaves, es decir, en Estudio, Facultad y Direccción.
Como hay 50 estudios diferentes, los valores de Estudio y Facultad se repiten en total 10.000-50=99.950 veces.
Como hay 10 facultades diferentes, el valor de Dirección se repite en total 10.000-10=99.990 veces.
El espacio desperdiciado por información redundante:
99.950*(50+50)+99.990*(100)= 19.994.000 bytes (asumiendo que cada carácter ocupa un byte).

c) (2 puntos) Implementar la detección de consistencia con respecto a una de las dependencias funcionales en PL/SQL.

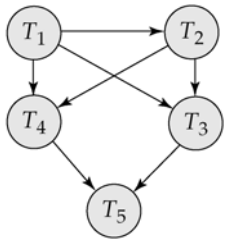
Solución (15 minutos):

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

```
CREATE OR REPLACE PROCEDURE ComprobarDF AS
  excepción_df EXCEPTION;
  v_Facultad "Alumnos".Facultad%TYPE;
  CURSOR c_Alumnos IS
    SELECT Facultad
    FROM
     (SELECT DISTINCT Facultad, Direccion
      FROM Alumnos)
    GROUP BY Facultad
    HAVING COUNT(Direccion)>1;
BEGIN
  OPEN c_Alumnos;
  LOOP
    FETCH c_Alumnos INTO v_Facultad;
```

```
    EXIT WHEN c_Alumnos%NOTFOUND;
    RAISE excepción_df;
  END LOOP;
  CLOSE c_Alumnos;
EXCEPTION
  WHEN excepción_df THEN
    DBMS_OUTPUT.PUT_LINE('A la facultad '||v_Facultad||' no le corresponde
siempre la misma dirección.');
    CLOSE c_Alumnos;
END;
```

3) (1,3 puntos) Considérese el siguiente grafo de precedencia:



Aplíquese el algoritmo del test de secuencialidad para determinar todas las planificaciones secuenciables.

Solución (10 minutos):

El grafo de secuencialización viene dado y no tiene ciclos. Por tanto, se calculan las secuencias de transacciones mediante ordenación topológica (que también nos asegurará que no haya ciclos) en el grafo:
1. Encontrar un nodo Ti sin arcos de entrada.
2. Listar Ti y eliminarlo.
3. Si quedan nodos, ir a 1.

1. T1
2. T2
3. Dos alternativas: T3 y T4. Escogemos T3:
4. T4
5. T5

Primera planificación secuenciable: T1, T2, T3, T4, T5
Si en el paso 3 escogemos T4 obtenemos la segunda y última planificación secuenciable: T1, T2, T4, T3, T5

4) (2,7 puntos) Considérese un índice B+ denso organizado sobre una clave sin repeticiones de 128 bytes de un fichero de datos de un millón de registros. El tamaño de bloque es de 1.024 bytes, las direcciones de bloque son de 4 bytes, la frecuencia de rotación del disco es de 7.200 rpm, el tiempo de búsqueda es de 10 ms, hay 128 sectores por pista y una pista por bloque. Suponiendo que cada nodo del árbol B+ se almacena en un bloque y que cada bloque del fichero de datos contiene sólo un registro de datos, se pide:
   a) (1,3 puntos) Calcular el grado máximo de salida (*n*) de los nodos, el espacio desperdiciado e indicar su estructura.

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:
$N*4*8+(N-1)*128*8+N-1 \leq 1.024*8$
$N*32+N*1.024-1024+N-1 \leq 8.192$
$N*(32+1.024+1) \leq 8.192+1.024+1$
$N \leq \lfloor 8,7 \rfloor = 8$, 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|

b) (1,4 puntos) Cuál es el tiempo de acceso a un dato usando y sin usar el fichero de índices?

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:
$\log_{8/2}(10^6) = 7$. El número total de valores es 10^6 (un millón de registros, no hay repetidos) y n=8.

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!