Segundo Parcial de Fundamentos de Bases de Datos
Noviembre 2018

SOLUCION

Duración: 3 horas y media
Presentar la resolución del parcial:

- Con las hojas numeradas y escritas de un solo lado. Comience cada ejercicio en una hoja nueva.
- Con la cantidad de hojas entregadas en la primer hoja.
- Con cédula de identidad y nombre en cada hoja.
- **Escrita a lápiz y en forma prolija.**

Ejercicio 1 (15 puntos)

Dado el esquema relación R(A,B,C,D,E,G,H) y el conjunto de dependencias F sobre R.
F = {A → DG , D → BA, G → C, CD → EH}
Para cada una de las siguientes afirmaciones indicar si son verdaderas o falsas. Justificar todas las respuestas:

a. (DG) es clave de R según F.

$(DG)^+$ = {D,G,B,A,C,E,H} = R por lo tanto DG es superclave, pero ¿es minimal?

$(D)^+$ = {D,B,A,G,C,E,H} = R, por lo que DG no es minimal y por lo tanto la afirmación es **FALSA**

b. A y D son las únicas claves de R según F.

Ya sabemos que D es clave, vemos si A también. Como A → DG podemos afirmar que $(D)^+=(A)^+$=R, entonces A también es clave.
¿hay más claves?
Sabemos que E,H,B no están en ninguna clave, porque sólo aparecen a la derecha en las dfs de F. Si hubieran otras claves, estarían contenidas en R-{ADEHB}={CG}.
Como $(CG)^+$={C,G} podemos afirmar que no hay más claves, y entonces la afirmación es **VERDADERA.**

c. Π_{R1} (F) = {G → C} siendo R_1 (C,D,G,H)

Como ya sabemos que D es clave, sabemos que D→CGH está en F^+ y se cumple en Π_{R1} (F), por lo tanto la afirmación es **FALSA**.

d. Sea F_1 = {A → D, A → G,D → B, D → A , G → C , CD → E, CD → H}. F_1 es un cubrimiento minimal de F

Como D es clave, C es redundante en CD → E y por lo tanto F_1 no es minimal. La afirmación es **FALSA**.

e. R está en 3NF respecto a F.

Todas las DF tienen superclaves de lado izquierdo, salvo G → C. Como C no es primo esta df viola 3NF y por lo tanto la afirmación es **FALSA.**

f. R no está en BCNF respecto a F.

Como R no está en 3NF tampoco puede estar en BCNF

g. La siguiente descomposición tiene JSP: R_1(A,H), R_1(B,C,D,E,G,H)

Por propiedad, la descomposición tiene JSP si y solo si $R_1 \cap R_2 \rightarrow R_1 - R_2$ o $R_1 \cap R_2 \rightarrow R_2 - R_1$ se cumplen en F^+
$R_1 \cap R_2$ = {H}, y como H^+={H} podemos afirmar que no se cumple ninguna de las dos dfs en F^+ y por lo tanto la afirmación es **FALSA**

Ejercicio 2 (15 puntos)

Dada la siguiente realidad:

En una agencia de viajes se quiere construir una base de datos con información acerca de las ventas de paquetes turísticos a clientes por parte de los vendedores de la agencia. Para esto se consideran los datos que se detallan a continuación.

Sobre los clientes se conoce su cédula de identidad, su nombre, fecha de nacimiento, dirección y teléfono. Cada paquete tiene un identificador, un destino principal, la aerolínea correspondiente al pasaje aéreo, la fecha de partida y un conjunto de nombres de hoteles (que son ofrecidos al cliente para que elija). Por otro lado, cada paquete incluye varios paseos. De cada paseo se conoce su nombre.

De los vendedores se conoce su cédula de identidad, su nombre, su teléfono y su fecha de ingreso a la agencia.

Cada vendedor puede realizar varias ventas de paquetes a varios clientes. En esta realidad, un mismo cliente puede realizar varias compras en la misma fecha, pero en una misma fecha es atendido por un sólo vendedor. Además un mismo cliente puede comprar un mismo paquete en varias fechas diferentes.

Utilizando los siguientes nombres para los atributos:

ci-cli	destino	ci-ven
nom-cli	aero	nom-ven
f-nac	f-part	tel-ven
dir-cli	nom-hotel	f-ing
tel-cli	nom-paseo	f-venta
paquete		

Se pide:

a) Deducir todas las dependencias funcionales que se cumplen.

ci-cli → nom-cli, f-nac, dir-cli, tel-cli

paquete → destino, aero, f-part

ci-ven → nom-ven, tel-ven, f-ing

ci-cli, f-venta → ci-ven

b) Para la relación universal R (tabla que contiene todos los atributos), y el conjunto de dependencias funcionales hallado en la parte a), hallar todas las claves. Justificar.

R (ci-cli, nom-cli, f-nac, dir-cli, tel-cli, paquete, destino, aero, f-part, nom-hotel, nom-paseo, ci-ven, nom-ven, tel-ven, f-ing, f-venta)

Los atributos: {nom-hotel, nom-paseo} no aparecen en ninguna dependencia funcional.

Los atributos: {f-venta, ci-cli, paquete} no aparecen a la derecha de ninguna dependencia funcional.

Entonces, los atributos: {nom-hotel, nom-paseo, f-venta, ci-cli, paquete} pertencen a TODAS las claves.

Hacemos la clausura transitiva de ese conjunto de atributos para ver si es clave.

(nom-hotel, nom-paseo, f-venta, ci-cli, paquete)+ =

{nom-hotel, nom-paseo, f-venta, ci-cli, paquete, nom-cli, f-nac, dir-cli, tel-cli, destino, aero, f-part, ci-ven, nom-ven, tel-ven, f-ing}

Es clave. Como todos esos atributos tienen que estar en todas las claves, no podemos generar ninguna otra clave que no sea superclave. Por lo tanto esta clave es única.

Clave única: (nom-hotel, nom-paseo, f-venta, ci-cli, paquete)

c) Dar una descomposición de R en 3NF sin pérdida de dependencias y con JSP. Aplicar el algoritmo dado en el curso (mostrando los pasos).

Paso 1 – Cubrimiento minimal

F = { ci-cli → nom-cli, f-nac, dir-cli, tel-cli

paquete → destino, aero, f-part

ci-ven → nom-ven, tel-ven, f-ing

ci-cli, f-venta → ci-ven }

– Separo atributos a la derecha de las dfs.

F1 = { ci-cli → nom-cli

ci-cli → f-nac

ci-cli → dir-cli

ci-cli → tel-cli

paquete → destino

paquete → aero

paquete → f-part

ci-ven → nom-ven

ci-ven → tel-ven

ci-ven → f-ing

ci-cli, f-venta → ci-ven }

– Elimino atributos redundantes a la izquierda de las dfs

La única df que podría tener un atributo redundante a la izquierda es la última. Debemos verificar si (ci-cli)+ contiene a ci-ven o (f-venta)+ contiene a ci-ven.

(ci-cli)+ = {ci-cli, nom-cli, f-nac, dir-cli, tel-cli}

(f-venta)+ = {f-venta}

Ni ci-cli ni f-venta es redundante en ci-cli, f-venta → ci-ven

– Elimino dfs redundantes

Si observamos las dfs de F1, vemos que en todas ellas el atributo que aparece a la derecha cumple que no aparece a la derecha de ninguna otra df. Por lo tanto, no es posible llegar a ninguno de ellos si elimino la df que lo contiene a la derecha.

Paso 2 – Generar relaciones a partir de las dfs. con igual lado izquierdo

R1 (ci-cli, nom-cli, f-nac, dir-cli, tel-cli)

R2 (paquete, destino, aero, f-part)

R3 (ci-ven, nom-ven, tel-ven, f-ing)

R4 (ci-cli, f-venta, ci-ven)

Paso 3 – Si la clave no se encuentra en ninguna de las relaciones generadas agregar una relación que la contenga

Agregamos:

R5 (nom-hotel, nom-paseo, f-venta, ci-cli, paquete)

Descomposición en 3NF con JSP y preservación de dfs:

ρ = {R1, R2, R3, R4, R5}

R1 (ci-cli, nom-cli, f-nac, dir-cli, tel-cli)

R2 (paquete, destino, aero, f-part)

R3 (ci-ven, nom-ven, tel-ven, f-ing)

R4 (ci-cli, f-venta, ci-ven)

R5 (nom-hotel, nom-paseo, f-venta, ci-cli, paquete)

d) Decir si la descomposición anterior está en BCNF, justificando. Si no está en BCNF, llevarla a dicha forma normal, aplicando el algoritmo dado en el curso (mostrando los pasos).

R1 (ci-cli, nom-cli, f-nac, dir-cli, tel-cli)

F_{R1} = {ci-cli $\rightarrow$ nom-cli, f-nac, dir-cli, tel-cli}

Clave única: ci-cli

R1 está en BCNF

R2 (paquete, destino, aero, f-part)

F_{R2} = {paquete $\rightarrow$ destino, aero, f-part}

Clave única: paquete

R2 está en BCNF

R3 (<u>ci-ven</u>, nom-ven, tel-ven, f-ing)

F_{R3} = {ci-ven → nom-ven, tel-ven, f-ing}

Clave única: ci-ven

R3 está en BCNF

R4 (<u>ci-cli, f-venta</u>, ci-ven)

F_{R4} = {ci-cli, f-venta → ci-ven}

Clave única: ci-cli, f-venta

R4 está en BCNF

R5 (nom-hotel, nom-paseo, f-venta, ci-cli, paquete)

F_{R5} = { }

Clave única: nom-hotel, nom-paseo, f-venta, ci-cli, paquete

R5 está en BCNF

La descomposición ρ está en BCNF.

e) Deducir todas las dependencias multivaluadas (no triviales) que se cumplen en cada tabla de la última descomposición obtenida. Para cada multivaluada que proponga, mostrar qué parte de la letra lo induce a pensar que ésta se cumple.

Existen multivaluadas no triviales solamente en R5, donde se cumplen:

M = { paquete ->> nom-hotel

paquete ->> nom-paseo }

Estas dependencias se deducen de la letra: “Cada paquete tiene un identificador, un destino principal, la aerolínea correspondiente al pasaje aéreo, la fecha de partida y un conjunto de nombres de hoteles... Por otro lado, cada paquete incluye varios paseos”

Para un paquete hay un conjunto de hoteles y un conjunto de paseos, conjuntos que son independientes entre sí. Por lo tanto, si están en una misma relación, para un mismo paquete deben combinarse todos los hoteles con todos los paseos.

Por otro lado, los atributos (f-venta, ci-cli) también deben combinarse con todos los hoteles y con todos los paseos. Esto lo muestran las multivaluadas que se generan **por complemento** de las anteriores (es decir, que se deducen de M):

paquete ->> nom-paseo, f-venta, ci-cli

paquete ->> nom-hotel, f-venta, ci-cli

Finalmente, se puede demostrar que del conjunto M también se deduce la siguiente multivaluada:

paquete ->> f-venta, ci-cli

f) Llevar la última descomposición obtenida a 4NF, aplicando el algoritmo dado en el curso (mostrando los pasos).

R1, R2, R3 y R4 están en 4NF porque no tienen dependencias multivaluadas no triviales. Debemos descomponer R5.

R5 (nom-hotel, nom-paseo, f-venta, ci-cli, paquete)

Mvds.: { paquete ->> nom-hotel

paquete ->> nom-paseo }

Tomo la mvd.: paquete ->> nom-hotel y genero la descomposición

R51 (paquete, nom-hotel)

Mvds: { } (Se cumple sólo la trivial)

R52 (paquete, f-venta, ci-cli, nom-paseo)

Mvd: paquete ->> nom-paseo

En R52 tomo la mvd.: paquete ->> nom-paseo y genero la descomposición

R521 (paquete, f-venta, ci-cli)

R522 (paquete, nom-paseo)

las cuales solo tienen mvds triviales.

Descomposición final de R5:

R51 (paquete, nom-hotel)

R521 (paquete, f-venta, ci-cli)

R522 (paquete, nom-paseo)

Descomposición de R en 4NF:

ρ = {R1, R2, R3, R4, R511, R512, R522}

Ejercicio 3 (15 puntos)

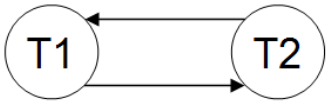
Considere las siguientes transacciones:

T1: r1(X) w1(X) r1(Y) w1(Y) r1(Z) w1(Z) c1

T2: w2(Z) r2(Y) w2(Y) c2

1. Considere la siguiente historia H1: w2(Z) r2(Y) r1(X) w1(X) r1(Y) w1(Y) w2(Y) c2 r1(Z) w1(Z) c1
Justificando cada una de sus respuestas, indique si la historia H1 cumple con alguna de las siguientes propiedades:

(a) **serializable:** No, porque el grafo de seriabilidad no es acíclico:



(b) **recuperable:** Si, porque T1 no confirma hasta que T2 (transacción desde la cual leyó T1), haya confirmado.
(c) **evita abortos en cascada:** Si, porque ninguna transacción lee de transacciones no confirmadas.
(d) **estricta:** No, porque T2 escribe el ítem Y antes de que T1 confirme.

2. Dar un historia entrelazada H2 de T1 y T2, cuyas transacciones sigan 2PL básico, pero que no sigan 2PL estricto. Utilizar bloqueos de escritura y de lectura.

A continuación, utilizando bloqueos de escritura y de lectura, T1 y T2 siguen 2PL básico pero no siguen 2PL estricto:

T1: rl1(X) r1(X) wl1(X) w1(X) rl1(Y) r1(Y) wl1(Y) w1(Y) rl1(Z) r1(Z) wl1(Z) w1(Z) u1(X) u1(Y) u1(Z) c1

T2: wl2(Z) w2(Z) rl2(Y) r2(Y) wl2(Y) w2(Y) u2(Z) u2(Y) c2

Y la historia H2 entrelazada de T1 y T2 es la que se presenta a continuación:

H2: rl1(X) r1(X) wl1(X) w1(X) wl2(Z) w2(Z) rl2(Y) r2(Y) wl2(Y) w2(Y) u2(Z) u2(Y) rl1(Y) r1(Y) wl1(Y) w1(Y) rl1(Z) r1(Z) wl1(Z) w1(Z) u1(X) u1(Y) u1(Z) c1 c2

3. ¿La historia H2 es serializable? **Justifique su respuesta**.

La historia es serializable porque toda historia formada por transacciones que siguen 2PL es serializable.

Ejercicio 4 (15 puntos)

Considere una porción de la base de datos de música construída en el laboratorio del curso, con el siguiente esquema relacional:

Temas (<u>tema</u>, titulo)
Contiene un conjunto de identificadores de temas musicales y sus títulos.

Lugares (<u>idLugar</u>, descLugar, pais)
Contiene información sobre los lugares. De cada lugar se conoce su identificador, su descripción y el país al que pertenece.

Compositores (<u>comp</u>, nombreComp, idLugar, nacionalidad)
Contiene información de los compositores de temas. De cada compositor se conoce su identificador, su nombre, el identificador de su lugar de nacimiento y su nacionalidad.

TemasComp (tema, comp)
Contiene los temas con los compositores que los crearon.

Además se cuenta con la siguiente información:

	Cant. Tuplas	Factor de bloqueo	Atributos	Índices
Temas	300000	150		Índice Primario sobre tema
Compositores	40000	80		Índice Primario sobre comp
TemasComp	630000	150		Índice Primario sobre <tema, comp>
Lugares	5000	100	Existen 100 países diferentes en la tabla, y se asume distribución uniforme.	Índice Primario sobre idLugar (cant niveles = 1) Índice Secundario sobre pais

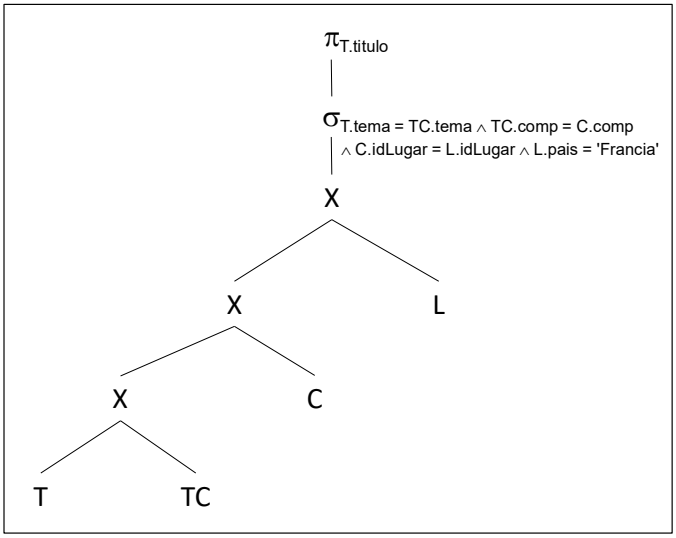
Considere la siguiente consulta SQL sobre esta base de datos:

```
SELECT   T.titulo
FROM     Temas T, TemasComp TC, Compositores C, Lugares L
WHERE    T.tema = TC.tema AND TC.comp = C.comp AND
         C.idLugar = L.idLugar AND L.pais = 'Francia'
```

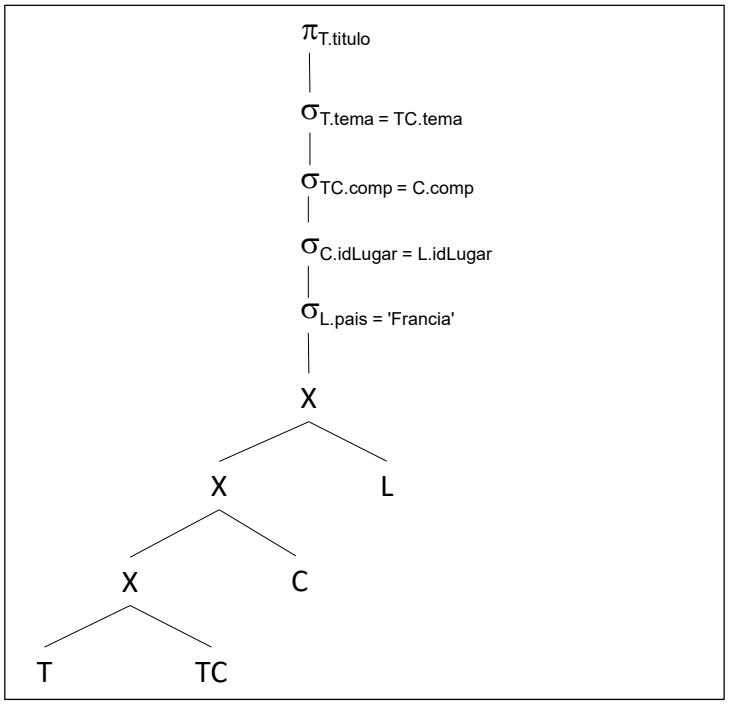
Se pide:

1. Partiendo del árbol canónico de la consulta, aplicar las heurísticas para obtener el plan lógico optimizado. Mostrar los pasos seguidos.

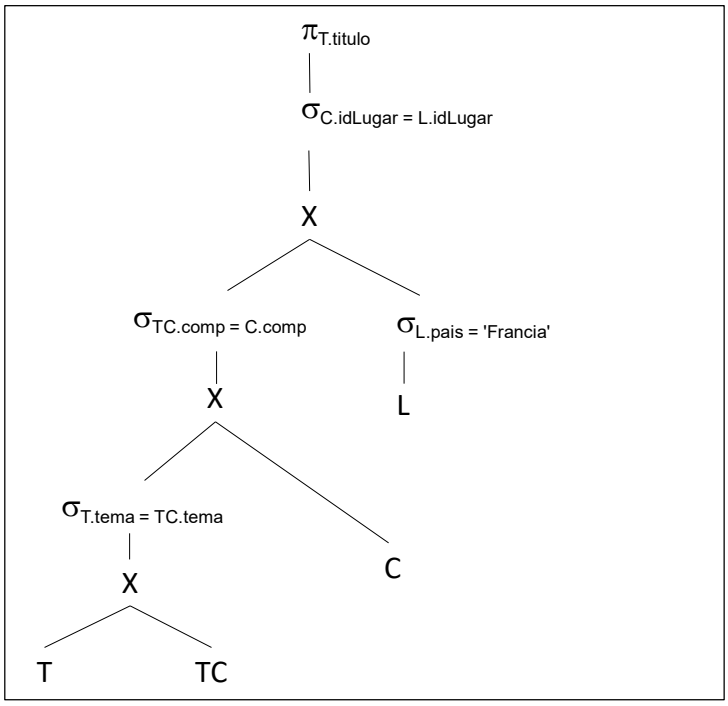
- Arbol Canónico



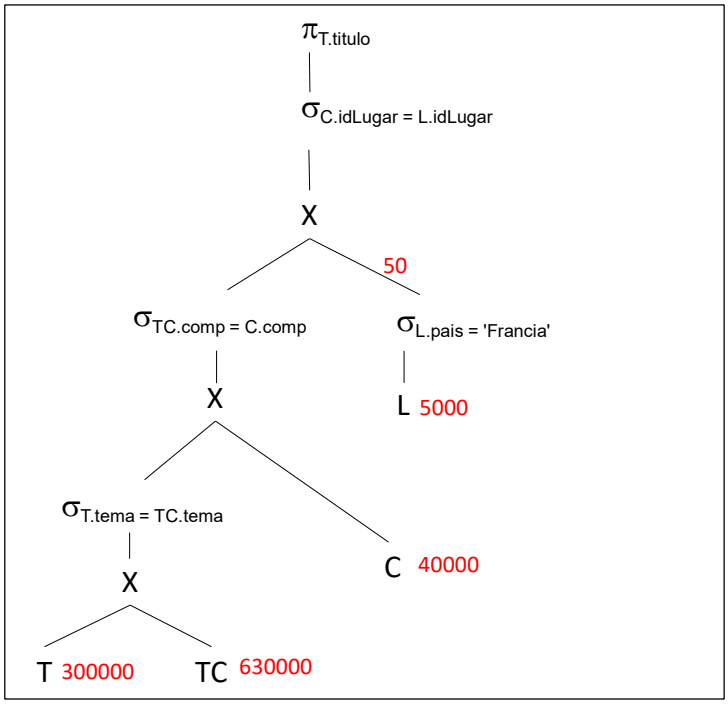
- Heurística 1

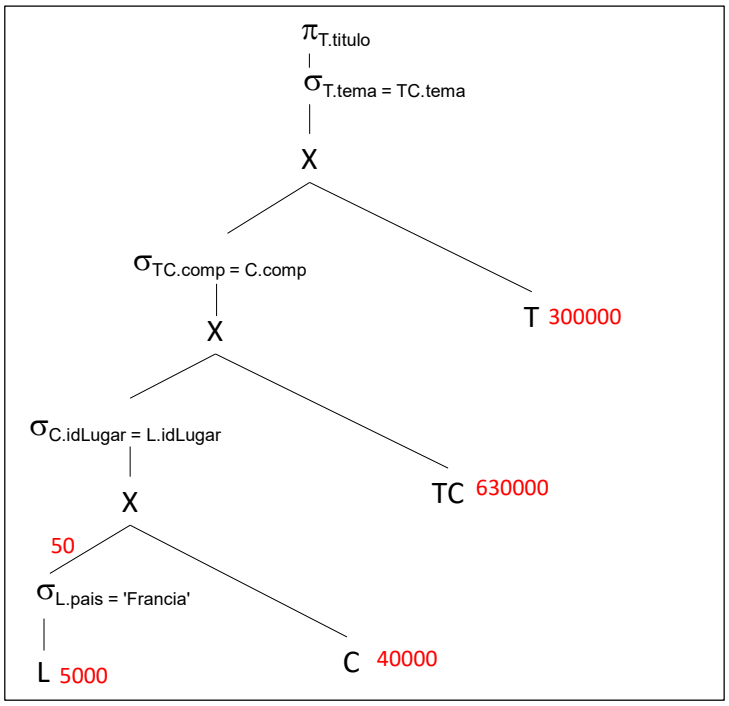


- Heurística 2

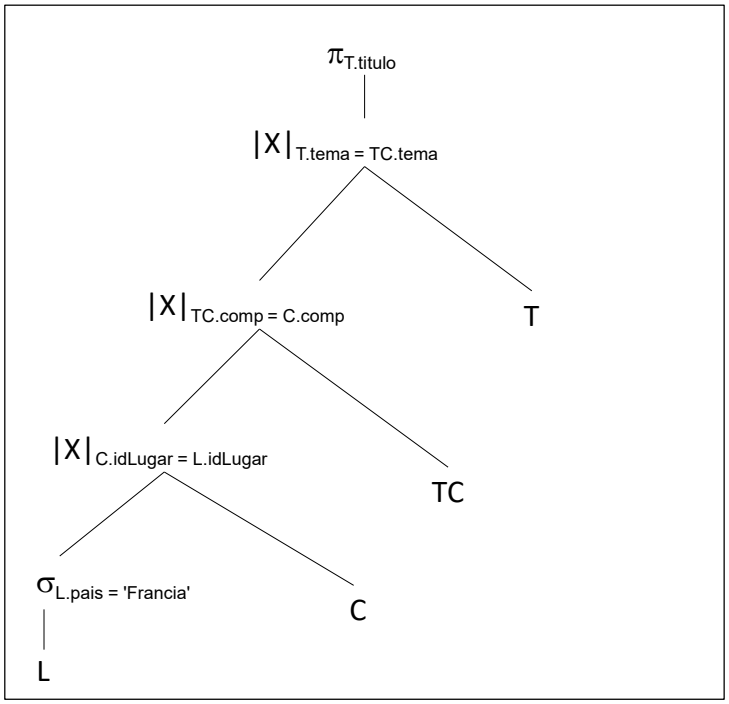


- Heurística 3

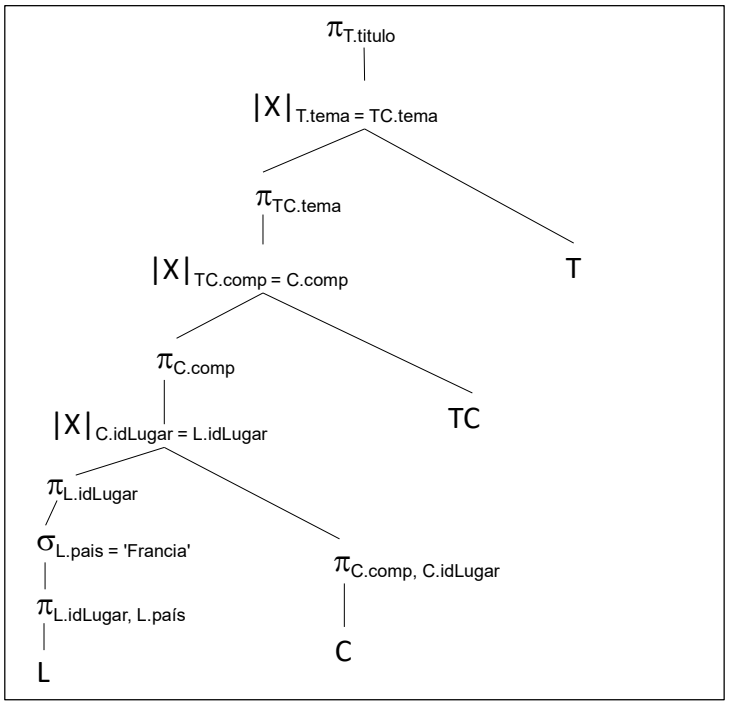




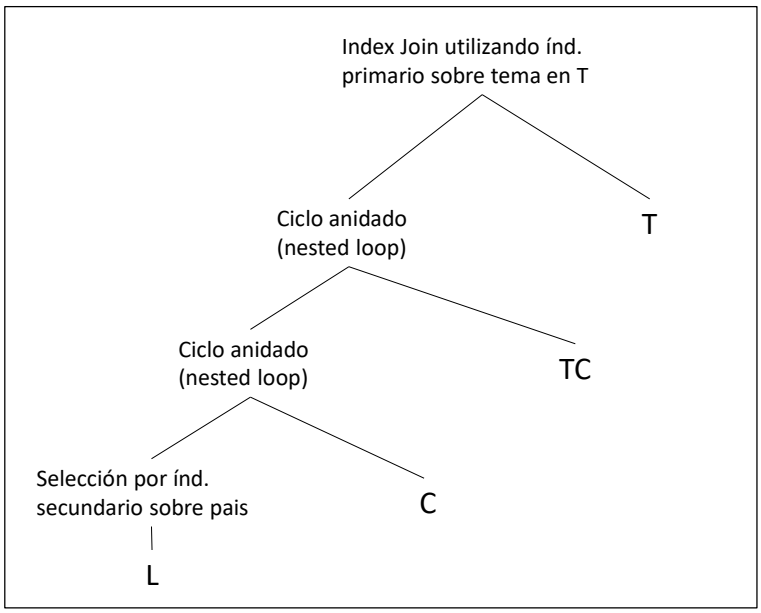
– Heurística 4



- Heurística 5



2. Dar un plan físico correspondiente al plan lógico anterior, donde se utilicen índices cuando sea posible.



3. Si existiera un índice secundario (arbol B+) en Compositores por el atributo idLugar, ¿se mejoraría el costo de la consulta? Justificar. (Sugerencia: Calcular solamente los costos de la parte del plan que se ve afectada)

Costo del join entre L y C, sin utilizar índice:

Costo (ciclo anidado por reg.) = $b_R + n_R * b_S$

tomando R = $\sigma_{L.pais = Francia}$ (Lugares) y S = Compositores

Costo (ciclo anidado por reg.) = $\lceil 50 / 100 \rceil + 50 * \lceil 40000 / 80 \rceil = 1 + 50 * 500 =$ **26000**

Costo (ciclo anidado por bloque) = $b_R + b_R / \lceil M-2 \rceil * b_S$

tomando R = $\sigma_{L.pais = Francia}$ (Lugares) y S = Compositores

Costo (ciclo anidado por bloque) = 1 + 1 * 500 = **501**

Costo del join entre L y C, utilizando índice secundario sobre idLugar en Compositores:

Costo (index join) = $b_R + n_R * (x + s_S)$

tomando R = $\sigma_{L.pais = Francia}$ (Lugares) y S = Compositores

Costo (index join) = 1 + 50 * (1 + 40000/100) = 1 + 50 * 401 = **20051**

Nota: Para el cálculo de s_S asumimos que en Compositores se encuentran todos los lugares y que hay distribución uniforme.

Respuesta:

Si se aplica el algoritmo de ciclo anidado por bloques, éste es menos costoso que el algoritmo que utiliza el índice. Por lo tanto, el utilizar el índice NO mejora el costo de la consulta.

Implementaciones de los Operadores.

	Algoritmo	Costo	Condición	Organización
$\sigma_c(R)$	Búsqueda Lineal	b_R (peor caso) $b_R/2$ (prom)	Todas	--
	Búsqueda Binaria	$\log_2 b_R + \lceil s/bf_R \rceil - 1$	Todas	Ordenado
	Indice Primario	x + 1	Igualdad	Ordenado
	Hash	1 o 2	Igualdad	--
	Índice Primario	x + (b/2) (prom)	de orden	Ordenado
	Índice Cluster	$x + \lceil s/bf_R \rceil$	Todas	Ordenado
	Índice secundario B+	x + s (peor caso)	Todas	--
	Grabación Intermedia	s/bf_R	Todas	--

	Algoritmo	Costo	Condición	Organización
$R \mid><\mid_c S$	Loop Anidado (registros)	$b_R+(n_R*b_s)$	Todas	--
	Loop Anidado (bloque)	$b_R+\lceil b_R/(M-2) \rceil * b_s$	Todas	--
	Sort Merge	$b_R + b_s$+ costo ordenación	Todas	Índice
	Index join	$b_R + n_R*Z$	Todas	Índice

Z depende del tipo de índice:

secundario: Z=x + sS

cluster: $Z=x + \lceil sS/bfS \rceil$

primario: Z=x+1