1. Consultas

1.1. Semántica de AR

Cuantificando sobre los dominios $\mathbb{D}_i$.

$$< R_i >= \mathbb{R}_i$$

$$< M\gamma N >=< M > \tilde{\gamma} < N >$$

$$< \sigma_{i\theta c}(M) >= \{< x_1 \dots x_k > \in < M > | x_i \tilde{\theta} c\}$$

$$< \sigma_{i\theta j}(M) >= \{< x_1 \dots x_k > \in < M > | x_i \tilde{\theta} x_j\}$$

$$< \Pi_{i_1 \dots i_n}(M) >= \{< x_{i_1} \dots x_{i_n} > | \exists x_{j_1} \dots x_{j_{(k-n)}},$$
$$\{i_1 \dots i_n, j_1 \dots j_{(k-n)}\} = \{1 \dots k\} < x_1 \dots x_k > \in < M >\}$$

1.2. Traducción AR a TRC

Invariante: la variable libre es t.

$$T'(E) = \{t | T_t(E)\}$$

$$T_t(R) = t \in R$$

$$T_t(R \cup S) = T_t(R) \vee T_t(S)$$

$$T_t(R \setminus S) = T_t(R) \wedge \neg T_t(S)$$

Recordemos que si la unión o diferencia se puede hacer en AR, es porque las relaciónes tienen aridades y tipos compatibles.

$$T_t(R \times S) = (\exists t_1, t_2) T_{t_1}(R) \wedge T_{t_2}(S) \wedge t[1] = t_1[1] \wedge \dots \wedge t[k] = t_1[k] \wedge t[k+1] = t_2[1] \wedge \dots \wedge t[k+n] = t_2[n]$$

$$T_t(\Pi_{i_1 \dots i_n}(R)) = (\exists t')(T_{t'}(R) \wedge t[1] = t'[i_1] \wedge \dots \wedge t[n] = t'[i_n])$$

$$T_t(\sigma_{i\theta c}(R)) = T_t(R) \wedge t[i] \ \tilde{\theta} \ \tilde{c}$$

$$T_t(\sigma_{i\theta j}(R)) = T_t(R) \wedge t[i] \ \tilde{\theta} \ t[j]$$

1.3. Semántica de TRC

$\mathbb{D}_1 \dots \mathbb{D}_r$ los dominios de los atributos.
Sea Tuples $= \bigcup(\mathbb{D}_{i_1} \times \dots \times \mathbb{D}_{i_l})$ con $l \leqslant r$ y $i_1 < \dots < i_l$.
v valuación, $v : \text{Var} \to \text{Tuples}$.

$$M = \{\mathbb{R}_1 \dots \mathbb{R}_n\} \models \phi[v]$$

Ahora obviemos el modelo explícito.

$$t \in R_i[v] \iff v(t) \in \mathbb{R}_i$$

$$(E_1 \wedge / \vee E_2)[v] \iff E_1[v] \ y/o \ E_2[v]$$

$$(\neg E)[v] \iff \text{no } E[v]$$

$$(\exists x)E[v] \iff \text{existe } a \in \text{Tuples}, E[v[a/x]]$$

$$(\forall x)E[v] \iff \text{para todo } a \in \text{Tuples}, E[v[a/x]]$$

Traducción de la consulta:

$$< \{t|\phi\} >= \{v(t)|M \models \phi[v]\}$$

1.3.1. Equivalencia

Veamos $\times$.
Queremos ver que:

$$\langle T'(R \times S)\rangle_{CT} = \langle R \times S\rangle_{AR}$$

Por definición de T':

$$\langle\{t|T_t(R \times S)\}\rangle_{CT}$$

Por definición de semántica de la consulta:

$$\{v(t)|M \models T_t(R \times S)[v]\}$$

Traducción:

$$\{v(t)|M \models ((\exists t_1)(\exists t_2)T_{t_1}(R) \wedge T_{t_2}(S) \wedge$$
$$t[1] = t_1[1] \wedge \dots \wedge t[k] = t_1[k] \wedge t[k+1] = t_2[1] \wedge \dots \wedge t[k+n] = t_2[n])[v]\}$$

Obviemos el $v(t)$ | como abuso de notación. Por semántica del $\exists$:

$$\text{existen } a_1, a_2(T_{t_1}(R) \wedge T_{t_2}(S) \wedge$$
$$t[1] = t_1[1] \wedge \dots \wedge t[k] = t_1[k] \wedge t[k+1] = t_2[1] \wedge \dots \wedge t[k+n] = t_2[n])[v[a_1/t_1][a_2/t_2]]$$

Por semántica del $\wedge$:

$$\text{existen } a_1, a_2 : T_{t_1}(R)[v[a_1/t_1][a_2/t_2]] \text{ y } T_{t_2}(S)[v[a_1/t_1][a_2/t_2]] \text{ y}$$
$$v(t)[1] = a_1[1] \text{ y } \dots \text{ y } v(t)[k] = a_1[k] y v(t)[k+1] = a_2[1] \text{ y } \dots \text{ y } v(t)[k+n] = a_2[n]$$

Por semántica buena onda:

$$\text{existen } a_1, a_2 : T_{t_1}(R)[v[a_1/t_1][a_2/t_2]] \text{ y } T_{t_2}(S)[v[a_1/t_1][a_2/t_2]] \text{ y}$$
$$v(t) \text{ es la concatenación de } a_1 \text{ y } a_2$$

Como t_1 es la única variable libre en $T_{t_1}(R)$ y análogamente para S:

$$\text{existen } a_1, a_2 : T_{t_1}(R)[a_1/t_1] \text{ y } T_{t_2}(S)[a_2/t_2] \text{ y}$$
$$v(t) \text{ es la concatenación de } a_1 \text{ y } a_2$$

Por definición de conjunto:

$$\text{existen } a_1, a_2 : a_1 \in \{a_1 | T_{t_1}(R)[a_1/t_1]\} \text{ y } a_2 \in \{a_2 | T_{t_2}(S)[a_2/t_2]\} \text{ y}$$
$$v(t) \text{ es la concatenación de } a_1 \text{ y } a_2$$

Por definición de $\times$ de conjuntos, y recordando que teníamos $v(t)$ |:

$$\{a_1 | T_{t_1}(R)[a_1/t_1]\} \times \{a_2 | T_{t_2}(S)[a_2/t_2]\}$$

Por hipótesis inductiva (notar que $a_i = w_i(t_i)$ donde $w_i = [a_i/t_i]$ por lo cual los conjuntos tienen la forma de la hipótesis):

$$\langle R \rangle_{AR} \times \langle S \rangle_{AR} = \langle R \times S \rangle_{AR}$$

Veamos $\setminus$.
Queremos ver que:

$$\langle T'(R \setminus S) \rangle_{CT} = \langle R \setminus S \rangle_{AR}$$

Por definición de T':

$$\langle \{t | T(R \setminus S)\} \rangle_{CT}$$

Por definición de semántica de la consulta:

$$\{v(t) | M \models T_t(R \setminus S)[v]\}$$

Traducción:

$$\{v(t) | M \models (T_t(R) \wedge \neg T_t(S))[v]\}$$

Por semántica del $\wedge$:

$$\{v(t) | T_t(R)[v] \text{ y } \neg T_{t_2}(S)[v]\}$$

Por semántica del $\neg$:

$$\{v(t) | T_t(R)[v] \text{ y no } T_{t_2}(S)[v]\}$$

Por teoría de conjuntos:

$$\{v(t) | T_t(R)[v]\} \cap \{v(t) | \text{ no } T_{t_2}(S)[v]\}$$

$$\{v(t) | T_t(R)[v]\} \cap \overline{\{v(t) | T_{t_2}(S)[v]\}}$$

$$\{v(t) | T_t(R)[v]\} \setminus \{v(t) | T_{t_2}(S)[v]\}$$

Por HI:

$$\langle R \rangle_{AR} \setminus \langle S \rangle_{AR} = \langle R \setminus S \rangle_{AR}$$

Veamos $\Pi_{i_1 \ldots i_n}$.
Queremos ver que:

$$\langle T'(\Pi_{i_1 \ldots i_n}(R)) \rangle_{CT} = \langle \Pi_{i_1 \ldots i_n}(R) \rangle_{AR}$$

Por definición de T':

$$\langle \{t | T_t(\Pi_{i_1 \ldots i_n}(R))\} \rangle_{CT}$$

Por definición de semántica de la consulta:

$$\{v(t) | M \models T_t(\Pi_{i_1 \ldots i_n}(R))[v]\}$$

Traducción:

$$\{v(t) | M \models ((\exists t_1) T_{t_1}(R) \wedge t[1] = t_1[i_1] \wedge \ldots \wedge t[k] = t_1[i_k])[v]\}$$

Obviemos el $v(t)$ | como abuso de notación. Por semántica del $\exists$:

$$\text{existe } a_1 : (T_{t_1}(R) \wedge t[1] = t_1[i_1] \wedge \ldots \wedge t[k] = t_1[i_k])[v[a_1/t_1]$$

Por semántica del $\wedge$:

$$\text{existe } a_1 : T_{t_1}(R)[v[a_1/t_1] \text{ y } v(t)[1] = a_1[i_1] \text{ y } \ldots \text{ y } v(t)[k] = a_1[k]$$

Por definición de conjunto:

$$\text{existe } a_1 : a_1 \in \{a_1 | T_{t_1}(R)[a_1/t_1]\} \text{ y } v(t)[1] = a_1[i_1] \text{ y } \ldots \text{ y } v(t)[k] = a_1[k]$$

Recordando que teníamos $v(t)$ |:

$$\{v(t) | \text{ existe } a_1 : a_1 \in \{a_1 | T_{t_1}(R)[a_1/t_1]\} \text{ y } v(t)[1] = a_1[i_1] \text{ y } \ldots \text{ y } v(t)[k] = a_1[k]\}$$

Por hipótesis inductiva:

$$\{v(t) | \text{ existe } a_1 : a_1 \in \langle R \rangle_{AR} \text{ y } v(t)[1] = a_1[i_1] \text{ y } \ldots \text{ y } v(t)[k] = a_1[k]\}$$

Reemplazando el a_1 por $< x_{j_1} \ldots x_{j_{(n-k)}} >$.

$$= \langle \Pi_{i_1 \ldots i_n}(R) \rangle_{AR}$$

1.3.2. División en AR

$$R = XY$$

$$S = Y$$

$$R \% S = \Pi_X(R) \setminus \Pi_X((\Pi_X(R) \times S) \setminus R)$$

1.4. SQL

1.4.1. División en SQL

$$R = A_1 \dots A_n$$

$$S = A_k \dots A_n, k \leqslant n$$

$R\,\%S$ = SELECT $A_1 \dots A_k$ FROM R R1
WHERE NOT EXISTS (SELECT $*$ FROM S S1
WHERE $R1.A_1, \dots, R1.A_{k-1}, S1.A_k, \dots, S1.A_n$ NOT IN R)

2. Optimización

2.1. Diseño físico

- T_R = # tuplas de R.
- B_R = # bloques de R.
- FB_R = factor de bloqueo de R.
- X = altura índice B^+.
- M = tamaño máximo de un bucket.
- T' = # tuplas de R que cumplen con la búsqueda.
- B' = # bloques de R que cumplen con la búsqueda.
- B_S = # bloques del índice que necesito recorrer para hacer el Index Scan.

	Heap file	Sorted file	B^+ clustered	B^+ unclustered	Hash
Exploración	B_R	B_R	B_R	B_R	B_R
=	B_R	$\log B_R + B'$	$X + 1 + B'$	$X + B_S + T'$	$T' + M$
Rango	B_R	$\log B_R + B'$	$X + 1 + B'$	$X + B_S + T'$	B_R

3. Normalización

3.1. Armstrong

3.1.1. Completitud

Lema: $X \to Y$ se sigue de F usando Armstrong $\Leftrightarrow Y \subseteq X^+$ donde $X^+ = \{A | X \to A$ se sigue por Armstrong de $F\}$. Sea $Y = A_1, ..., A_n$.

$\Rightarrow$) Por descomposición. $X \to A_i$ se sigue de $F \Rightarrow A_i \in X^+$.

$\Leftarrow$) $\forall i, X \to A_i$ se sigue de F (por def de X^+) $\Rightarrow$por unión, $X \to Y$.

Quiero ver que $F \models X \to Y \Rightarrow X \to Y$ se deriva por Armstrong de F.

Sea $X \to Y$ tal que no se deriva de F por Armstrong y sea r:

X^+	Otros
$1 \dots 1$	$1 \dots 1$
$1 \dots 1$	$0 \dots 0$

Veamos primero r cumple F: si $V \to W$ no se cumple, $V \subseteq X^+$ y $\exists A, A \in W, A \notin X^+$ (por definición). Ahora, por lema $X \to V$ se sigue por Armstrong y $V \to W \in F$ asi que por transitividad $X \to W$ se sigue por Armstrong, entonces $X \to A$ se sigue por Armstrong $\Rightarrow A \in X^+$ por definición, absurdo.

Si $X \to Y$ fuera satisfecho por r, como $X \subseteq X^+ \Rightarrow Y \subseteq X^+$ (sino no se cumpliría por la forma de r) $\Rightarrow$por lema que $X \to Y$ se deriva de F por Armstrong, absurdo.

3.2. Definiciones

3.2.1. Partición sin pérdida de información por junta

R esquema, F conjunto de dependencias, $\rho = \{R_1, \ldots, R_k\}$ partición es **sin pérdida de información por junta** sii:

$$r \in R \Rightarrow r = \bowtie_{i=1}^{k} \Pi_{R_i}(r)$$

3.2.2. Partición sin pérdida de dependencias funcionales

R esquema, F conjunto de dependencias, $\rho = \{R_1, \ldots, R_k\}$ partición.

$$\Pi_{Z \subseteq X}(F) = \{(X \to Y) \in F^+ | XY \subseteq Z\}$$

ρ es **sin pérdida de dependencias** sii:

$$F^+ = (\cup_{i=1}^{k}(\Pi_{R_i}(F)))^+$$

3.2.3. 3NF

Un atributo es **primo** si pertenece a alguna clave.
Un esquema está en **3NF** sii $\forall (X \to Y) \in F^+, A \notin X \Rightarrow A$ es superclave o A es primo.

3.2.4. BCNF

Un esquema está en **BCNF** sii $\forall (X \to Y) \in F^+, A \notin X \Rightarrow A$ es superclave.

3.3. Algoritmos

3.3.1. Clausura de un conjunto de atributos

Con respecto al conjunto de dependencias F.

$$X^0 = X$$

$$X^{i+1} = X^i \cup \{A | A \in Y, Z \to Y \in F, Z \in X^i\}$$

$$X_F^+ = X^j \iff X^j = X^{j+1}$$

3.3.2. Correctitud

Dado que U el universo de atributos es finito, y que $X^k \subseteq X^{k+1} \subseteq U \forall k$, el algoritmo tarde o temprano se estabiliza. Sea i el índice para el cual eso sucede. Veamos que $X^i = X_F^+$.
$X^i \subseteq X_F^+$. Veremos que en cada paso se agregan cosas que están en la clausura.

Caso base: $j = 0$. $X^0 = X$ y por lo tanto si $A \in X^0, A \in X$ y entonces vale $X \to A$ por reflexividad, y $A \in X_F^+$.

Paso inductivo: $j > 0$, $X^{j-1} \subseteq X_F^+$ por HI. Si A se agrega en el paso j, hay $Y \to Z \in F$, $Y \subseteq X^{j-1}$ y $A \in Z$. $Y \subseteq X_F^+$, y por lo tanto, $A \in X_F^+$:

$Y \subseteq X_F^+ \Rightarrow X \to Y$
$X \to Y, Y \to Z \Rightarrow X \to Z$ por transitividad.
$X \to Z \Rightarrow X \to A$ por descomposición.

$X \to A \Rightarrow A \in X_F^+$ por definición.

$X_F^+ \subseteq X^i$. Supongamos que no. $\exists A \in X_F^+, A \notin X^i$. Sea r:

X^i	Otros
$1 \dots 1$	$0 \dots 0$
$1 \dots 1$	$1 \dots 1$

r satisface F pues si no, $\exists U \to V, U \in X^i, V \cap (U \setminus X^+) \neq \emptyset$, y debería haber agregado más cosas a X^i, lo cuál es absurdo.

$A \in X_F^+ \Rightarrow F \models X \to A$. Como r cumple F, $X \to A$ tiene que valer en r. Dado que $X \in X^i$, si $A \notin X^i$, $X^{i+1} \neq X^i$, lo cual es absurdo.

3.3.3. Claves

R esquema, F conjunto de dependencias.

- Pongo X el conjunto de atributos que no aparece en ningún lado derecho de ninguna dependencia.
- Si $X_F^+ = R$, ya estoy.
- Si no, voy agregando atributos y clausurando, primero de a uno, y así.
- Si el conjunto inicial fuera vacío, tengo que probar desde el vamos.

3.3.4. Cubrimiento minimal

- $X \to YZ \Rightarrow X \to Y \wedge X \to Z$
- $XZ \to Y \wedge Y \in X^+ \Rightarrow X \to Y$
- $X \to Y \in F, (F \setminus \{X \to Y\})^+ \equiv F^+ \Rightarrow F \setminus \{X \to Y\}$
- (Equivalente al anterior) $Y \in X^+_{F \setminus \{X \to Y\}} \Rightarrow F \setminus \{X \to Y\}$

3.3.5. Pérdida por junta - Partición binaria

R esquema, F conjunto de dependencias, $\rho = (R_1, R_2)$ partición.
ρ es SPI sii:

$$F \vdash (R_1 \cap R_2) \to (R_1 \setminus R_2)$$

ó

$$F \vdash (R_1 \cap R_2) \to (R_2 \setminus R_1)$$

3.3.6. Correctitud

Queremos ver que $F \vdash (R_1 \cap R_2) \to R_1 \setminus R_2 \Rightarrow \rho$ SPI.
Supongamos $F \vdash (R_1 \cap R_2) \to R_1$.
Veremos $\forall r, r \models F \Rightarrow r = r_1 \bowtie r_2$.
Recordemos $r_i = \Pi_{R_i}(r)$.
$r \subseteq r_1 \bowtie r_2$ por definición de junta natural.
Ahora veremos $r_1 \bowtie r_2 \subseteq r$:

$$t \in r_1 \bowtie r_2 \Rightarrow \exists t_1, t_2 \in r, t_1[R_1] = t[R_1], t_2[R_2] = t[R_2], t_1[R_1 \cap R_2] = t_2[R_1 \cap R_2] \Rightarrow t_1[R_1] = t_2[R_1] \Rightarrow t \in r$$

Queremos ver que ρ SPI $\Rightarrow F \vdash (R_1 \cap R_2) \rightarrow R_1 \setminus R_2$.

Veamos primero que cualquier atributo que termina en una columna de a pertenece a $(R_1 \cap R_2)^+$.

Supongamos que no. Dado que solamente se ponen a, sea k la mínima cantidad de pasos que corre el algoritmo hasta poner a en la columna de algún atributo que no está en $(R_1 \cap R_2)^+$. Sea A ese atributo.

Si en el paso k se cambia A a a, sucede que existe $U \rightarrow V \in F$, tal que $A \in V$, y todas las columnas correspondientes a atributos de U están en a. Dado que A es el primero que no está en $(R_1 \cap R_2)^+$ en ponerse en a, $U \subseteq (R_1 \cap R_2)^+$. Eso implica que $F \vdash (R_1 \cap R_2)^+ \rightarrow U$. Sin embargo, eso y transitividad implican $F \vdash (R_1 \cap R_2)^+ \rightarrow A$ con un paso de descomposición; absurdo pues supusimos $A \notin (R_1 \cap R_2)^+$.

Como supusimos que ρ era SPI, hay una fila que queda en a. Por el lema anterior, todos esos atributos están en $(R_1 \cap R_2)^+$. Sea R_i esa fila. Por lo tanto, podemos concluir $F \vdash (R_1 \cap R_2) \rightarrow R_i \setminus R_{(i+1) mod 2}$.

3.3.7. Pérdida por junta - Tableau

Las columnas de la matriz son los atributos de R y las filas son los elementos de la partición. Luego, si el atributo A_j está en el subesquema R_i, escribo a_j en la posición (i, j) de la matriz. Si no, escribo b_{ij}.

Mientras haya cambios en la matriz, para cada dependencia, si la matriz tiene valores similares en el lado izquierdo de la dependencia, igualo los lados derechos:

- a_j gana sobre b_{ij}, y reemplazo **todas** las ocurrencias de b_{ij}.
- Dos elementos del mismo tipo se igualan en algún sentido y se reemplaza en todas las ocurrencias.

3.3.8. Correctitud

$$m_\rho(r) = \bowtie_{i=1}^{k} \Pi_{R_i}(r)$$

Supongamos que no hay filas con todas a's. Podemos ver a la tabla como una relación r para el esquema R, con tantas tuplas como filas en la matriz.

r satisface F porque así fue construída. Veremos que $r \neq m_\rho(r)$.

Claramente $(a_1 \dots a_n \notin r$. Sin embargo, si llamamos μ_i a la i-ésima fila, $\mu_i[R_i] = a_i$. Por lo tanto, $(a_1 \dots a_n) \in m_\rho(r)$, QED.

3.3.9. 3NF sin pérdida (a.k.a. por síntesis)

R esquema, F minimal cover.

- Poner todos los atributos que no están en ninguna dependencia en un nuevo subesquema y quitarlos de R.
- Si alguna dependencia incluye todos los atributos de R, devolver R.
- Para cada d.f. $X \rightarrow A$ en F, poner XA como nuevo subesquema.

Esto es SPD.

Si a esa descomposición le agrego una clave en un esquema único, queda SPI.

3.3.10. Descomposición binaria a BCNF

R esquema.

Para cada $X \rightarrow Y$ que viola BCNF, XY y $R \setminus \{Y\}$ como subesquemas, e itero en el segundo subesquema.

3.3.11. Algoritmo pérdida de dependencias

Quiero ver si $X \to Y \in (\cup_{i=1}^{k}(\Pi_{R_i}(F)))^+$.

- $Z = X$
- Mientras haya cambios en Z
- Para cada elemento R_i de la partición
- $Z = Z \cup ((Z \cap R_i)_F^+ \cap R_i)$

Si $Y \in Z$ entonces $X \to Y \in (\cup_{i=1}^{k}(\Pi_{R_i}(F)))^+$.

4. Transacciones

4.1. Locking binario

4.1.1. Definición

- LOCK: permite lectura y escritura
- UNLOCK: libera el item

4.1.2. Semántica

Se asocia una función f a cada par LOCK/UNLOCK. La función representa la operación realizada por ese fragmento de la transacción sobre el ítem bloqueado, y dependiente de los ítems leídos hasta el momento.

Los ítems empiezan con un valor I_0. Dos schedules son equivalentes si al finalizar ambos, cada item es el resultado de una composición de funciones *sintácticamente* equivalentes.

4.1.3. Test de serializabilidad

Schedule S. Grafo dirigido $G =< V, X >$.
$v_i \in V \iff T_i \in S$.
$(v_i, v_j) \in X \iff i \neq j, T_i$: UNLOCK A_m ... T_j : LOCK A_m en S y j es la primera transacción en lockear.

Si el grafo tiene ciclos, S no es serializable. En caso contrario, el schedule serial equivalente es el ordenamiento topológico de los nodos.

4.1.4. Correctitud del test

Sea G el grafo de S. Supongamos que G tiene un ciclo:
$T_{j_1} \to \ldots \to T_{j_n} \to T_{j_1}$.

Sea R equivalente a S, R serial. Sea T_{j_p} la primera transacción del ciclo en R. Seguro que en S están $T_{j_{p-1}} \to T_{j_p}$ por el item A, con $p - 1 = n \iff p = 1$.

En R, como T_{j_p} aparece antes que $T_{j_{p-1}}$ (porque es la primera), la función final para A tiene seguro g de Tj_{p-1} aplicada a f de T_{j_p}. En S es exactamente al revés, y por lo tanto S no es equivalente a ningún schedule serial (pues no supusimos nada especial sobre R).

Supongamos que el grafo no tiene ciclos. Sea la profundidad de una transacción la posición en el ordenamiento topológico. Veremos por inducción en la profundidad que cada transacción lee la misma función de cada ítem tanto en S como en R (la resultante del ordenamiento topológico).

Supongamos que esto no sucede. En S, T lee de T', y en R lee de T''.

Sean $T_{i_1} \ldots T_{i_r}$ la secuencia de transacciones que lockean A en S. Claramente en esta secuencia T' precede a T. T'' no puede estar en el medio, pues dado que también lockea A, está en algún otro lugar de la secuencia. Entonces, en realidad, en el órden topológico, T también lee de T'.

Finalmente, por hipótesis inductiva, podemos concluir que T' lee correctamente, y por lo tanto T también. El caso base es trivial pues la primera transacción lee los valores iniciales en ambos casos (pues si es la primera es justamente porque no le llegan flechas).

4.2. 2PL y locking binario

2PL con locking binario da siempre schedules serializables. Supongamos que no. Existe entonces un ciclo:

$T_{j_1} \to \ldots \to T_{j_n} \to T_{j_1}$.

Cada flecha está dada por un UNLOCK de T_{j_k} y un LOCK de $T_{j_{k+1}}$. Por lo tanto, hay un UNLOCK de T_{j_1} que precede a un LOCK de T_{j_2}. Ambos dos preceden a un UNLOCK de T_{j_n}, que a su vez precede a un LOCK de T_{j_1}. Pero esto es absurdo, pues entonces hay un UNLOCK que precede a un LOCK en T_{j_1}, y eso violaría 2PL.

4.3. 2PL como protocolo más optimista

Si T_1 *no* es 2PL, entonces es de la forma:

$\vdots$
UNLOCK A
$*$
LOCK B
$\vdots$

Sea T_2:
LOCK A
LOCK B
UNLOCK A
UNLOCK B

Si T_2 es ejecutada entera en $*$, el grafo tiene un ciclo.