Control de Concurrencia Multiversión

Dr. Gerardo Rossel

FCEN, UBA

2do Cuat. 2025

Timestamping Multiversión

Definición

- Variación/Extensión del planificadores monoversión
- Mantiene versiones históricas de los items
- Permite que las transacciones lean valores antiguos
- **Evita** aborts ocasionados por eventos **read-too-late**

Multiversión

Lectura y Escritura

Una operación de lectura de la forma $r(x)$ **lee una versión existente** de x, y una operación de escritura de la forma $w(x)$ (siempre) **crea** una nueva versión de x **o sobrescribe** una existente.

Asumimos que cada transacción escribe cada elemento de datos como máximo una vez; por lo tanto, si t_j contiene la operación $w_j(x)$, podemos denotar la versión de x creada por esta escritura como x_j.

Motivación

$$s = r_1(x)\ w_1(x)\ r_2(x)\ w_2(y)\ r_1(y)\ w_1(z)\ c_1\ c_2 \qquad \rightarrow \notin \text{CSR}$$

pero el schedule sería tolerable si
$r_1(y)$ pudiera leer la versión y_0 de y
para ser consistente con $r_1(x)$
$\rightarrow$ s sería equivalente al serial $s' = t_1 t_2$

Enfoque

- Cada paso de escritura w crea una nueva versión
- Cada paso de lectura r podría elegir qué versión quiere/necesita leer
- Las versiones son transparentes para la aplicación y transitorias (es decir, sujetas a recolección de basura)

Sobre Historias y Schedules

Sea $T = \{t_1, \ldots, t_n\}$ un conjunto de transacciones, donde cada $t_i \in T$ tiene la forma $t_i = (op_i, <_i)$ con op_i denotando las operaciones de t_i y $<_i$ su orden.

Una **historia** para T es un par $s = (op(s), <_s)$ tal que:

(a) $op(s) \subseteq \bigcup_{i=1}^{n} op_i \cup \bigcup_{i=1}^{n} \{a_i, c_i\}$

(b) para todo i, $1 \leq i \leq n$: $c_i \in op(s) \iff a_i \notin op(s)$

(c) $\bigcup_{i=1}^{n} <_i \subseteq <_s$

(d) para todo i, $1 \leq i \leq n$, y todo $p \in op_i$: $p <_s c_i$ o $p <_s a_i$

(e) para todo $p, q \in op(s)$ tal que al menos uno de ellos es una escritura y ambos acceden al mismo elemento de datos: $p <_s q$ o $q <_s p$

Un **schedule** es un prefijo de una historia

Función de Versión

Definición

Sea s una historia. Una **función de versión** para s es una función h, que:

- Asocia cada operación de lectura con una operación de escritura anterior del mismo ítem
- Para operaciones de escritura es la identidad

1. $h(r_i(x)) = w_j(x)$ para algún $w_j(x) <_s r_i(x)$, y $r_i(x)$ lee x_j ,
2. $h(w_i(x)) = w_i(x)$, y $w_i(x)$ escribe x_i .

Historia Multiversión

Una historia multiversión (**mv**) para las transacciones $T = \{t_1, \ldots, t_n\}$ es un par $m = (op(m), <_m)$ donde $<_m$ es un orden en $op(m)$ y

(a) $op(m) = \bigcup_{i=1}^{n} h(op(t_i))$ para alguna función de versión h,

(b) para todo $t \in T$ y todo $p, q \in op(t)$: $p <_t q \Rightarrow h(p) <_m h(q)$,

(c) si $h(r_j(x)) = r_j(x_i)$, $i \neq j$, entonces c_i está en m y $c_i <_m c_j$.

Un **schedule** multiversión (mv) es un prefijo de una historia multiversión.

Historia Multiversión

Schedule Multiversión

Una historia multiversión contiene operaciones de lecturas y escrituras versionadas para sus transacciones y el orden de las mismas respeta el orden de las tranasacciones individuales

Condición importante sobre Commits

Si $h(r_j(x)) = r_j(x_i), i \neq j$, y c_j está en m, entonces c_i está en m y $c_i <_m c_j$

Un planificador multiversión es un **planificador monoversión** si su función de versión asigna cada lectura a la última escritura precedente en el mismo elemento de datos.

Serializabilidad Multiversión

- View Serializabilidad
- Conflicto Serializabilidad

Leer de (Reads-From Relation)

Sea m un schedule multiversión, t_i y $t_j \in trans(m)$ La relación *lee-de* se define cómo: $RF(m) := (t_i, x, t_j) | r_j(x_i) \in op(m)$

Sean m y m' dos schedules multiversión tales que $trans(m') = trans(m)$ entonces m y m' son view equivalentes $(m \approx_v m')$ si $RF(m) = RF(m')$

Serializabilidad Multiversión

Multiversión view serializable

Sea m una historia multiversión, se dice que m es multiversión view serializable si existe una historia m' serial monoversión tal que $m \approx_v m'$.

MVSR es la clase de historias *view* serializables (serializables por vista) multiversión.

Decidir si una historia esta en MSVR es un problema NP-Completo

Grafo de Conflictos - RF

El grafo de conflictos de una historia m denotado como $G(m)$ se construye con nodos por cada transacción de m con un eje $t_i \rightarrow t_j$ si $r_j(x_i)$ esta en m
Para cualquier par de schedulers multiversión, $m \approx_v m'$ **entonces** $G(m) = G(m)'$.

Grafo de Conflictos - RF

El grafo de conflictos de una historia m denotado como $G(m)$ se construye con nodos por cada transacción de m con un eje $t_i \rightarrow t_j$ si $r_j(x_i)$ esta en m
Para cualquier par de schedulers multiversión, $m \approx_v m'$ **entonces** $G(m) = G(m)'$.

$m = w_1(x_1)r_2(x_0)w_1(y_1)r_2(y_1)c_1c_2$
$m' = w_1(x_1)w_1(y_1)c_1r_2(x_1)r_2(y_0)c_1$

$G(m) = G(m)'$ pero $m \not\approx_v m'$

Grafo de Serialización Multiversión (MVSG)

Definición: Un orden de versiones para un ítem de datos x, denotado $\ll_x$, es un orden total entre todas las versiones de x.

Un orden de versiones para un plan multiversión m es la unión de los órdenes de versiones para todos los ítems escritos en m.

Dado un plan m y un orden de versiones $\ll$, el Grafo de Serialización Multiversión, $MVSG(m, \ll)$, es un grafo dirigido donde:

- Los nodos son las transacciones en m
- Las aristas se definen por las reglas de conflicto y de orden de versiones

Aristas en $MVSG(m, \ll)$

$MVSG(m, \ll)$ contiene las siguientes aristas:

(i) Todas las aristas del grafo de conflictos $G(m)$. (Para $r_k(x_j) \in op(m)$, agregar arista $t_j \rightarrow t_k$)

(ii) Para $r_k(x_j), w_i(x_i) \in op(m)$: Si $x_i \ll x_j$, entonces agregar arista $t_i \rightarrow t_j$.

(iii) Para $r_k(x_j), w_i(x_i) \in op(m)$: Si $x_j \ll x_i$, entonces agregar arista $t_k \rightarrow t_i$.

Idea clave: El *MVSG* extiende el grafo de conflictos incorporando dependencias de orden de versiones.

Un historial multiversión m pertenece a MVSR (Multiversion Serializability) **si y solo si** existe un orden de versiones $\ll$ tal que el Grafo de Serialización Multiversión $MVSG(m, \ll)$ sea **acíclico**.

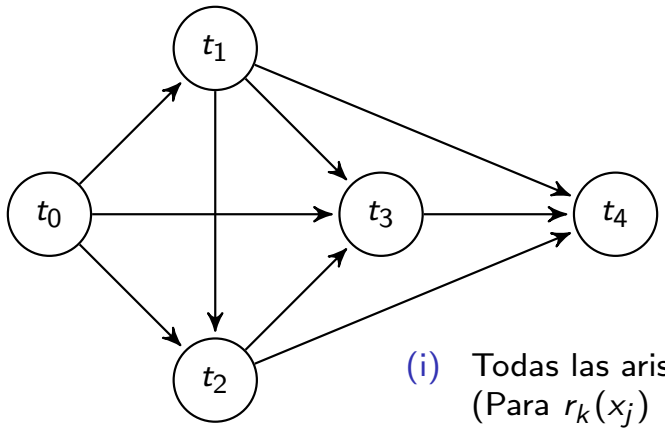
Ejemplo MVSG

Historial multiversión (schedule m):

$$m = w_0(x_0)\, w_0(y_0)\, w_0(z_0)\, c_0\, r_1(x_0)\, r_2(x_0)\, r_2(z_0)\, r_3(z_0)$$
$$w_1(y_1)\, w_2(x_2)\, w_3(y_3)\, w_3(z_3)\, c_1\, c_2\, c_3\, r_4(x_2)\, r_4(y_3)\, r_4(z_3)\, c_4$$

Orden de versiones:

$$x_0 \ll x_2, \quad y_0 \ll y_1 \ll y_3, \quad z_0 \ll z_3$$



(t_1, t_2) : $r_1(x_0), w_2(x_2) \in op(m)$, $x_0 \ll x_2$

(t_1, t_3) : $w_1(y_1), r_4(y_3) \in op(m)$, $y_1 \ll y_3$

(t_2, t_3) : $r_2(z_0), w_3(z_3) \in op(m)$, $z_0 \ll z_3$

(i) Todas las aristas del grafo de conflictos $G(m)$. (Para $r_k(x_j) \in op(m)$, agregar arista $t_j \rightarrow t_k$)

(ii) Para $r_k(x_j), w_i(x_i) \in op(m)$: Si $x_i \ll x_j$, entonces agregar arista $t_i \rightarrow t_j$.

(iii) Para $r_k(x_j), w_i(x_i) \in op(m)$: Si $x_j \ll x_i$, entonces agregar arista $t_k \rightarrow t_i$.

MCSR

MCSR

Vamos a trabajar con una subclase de MVSR cuya pertenencia puede determinarse en tiempo polinómico.

Conflicto Serializable

Multiversión conflicto serializable

Un conflicto multiversión en un schedule multiversión m es un par de pasos $r_i(x_j)$ y $w_k(x_k)$ tales que $r_i(x_j) <_m w_k(x_k)$

MCSR

- El único tipo relevante de conflictos son los pares de operaciones *read-write* en el mismo ítem de datos, no necesariamente en la misma versión
- Los pares *write-write* en el mismo ítem de datos ya no se consideran conflictos, ya que crean versiones diferentes y depende de las operaciones de lectura elegir la versión adecuada
- ¿Qué pasa con los pares *write-read*? La esencia de un conflico pq es que no pueda permutarse p y q

Conmutatividad

Pasos de transformación

Un paso de transformación intercambia el orden de dos pasos adyacentes (es decir, pasos p, q con $p < q$ tal que $o < p$ y $q < o$ para todos los demás pasos o) pero sin invertir el orden de un par de conflicto multiversión (es decir, *rw*).

Reducibilidad multiversión

Una historia multiversión es *multiversión reducible* si puede **transformarse** en una *historia serial monoversión* mediante una secuencia de pasos de transformación.

Multiversión conflicto serializable

Un historia multiversión m es multiversión conflicto serializable si hay una historia monoversión serial para el mismo conjunto de transacciones en el que todos los pares de operaciones en conflicto multiversión ocurren en el mismo orden que en m.

MCSR denota la clase de todas las historias multiversión conflicto serializables.

Teorema

Una historia multiversión es multiversión reducible **si y solo si** es multiversión conflicto serializable,

$$MCSR \subset MVSR$$

Grafo de conflicto multiversión

Sea m un schedule de multiversión. El *grafo de conflicto multiversión* de m es un grafo que tiene las transacciones de m como sus nodos y una arista de t_i a t_k si existen pasos $r_i(x_j)$ y $w_k(x_k)$ para el mismo elemento de datos x en m tal que $r_i(x_j) <_m w_k(x_k)$.

Una historia multiversión es MCSR si y solo si su grafo de conflictos de multiversión es acíclico

$m \in MCSR \iff$ multiversión reducible *iff* grafo conflicto multiversión es acíclico.

Atención

La historia monoversión serial equivalente para un grafo de conflictos multiversión acíclico no se puede derivar simplemente ordenando topológicamente el grafo

La aciclicidad del grafo es una condición suficiente y necesaria para la existencia de una secuencia de pasos de transformación (en el sentido de la definición de reducibilidad multiversión) que produzca una historia monoversión serial