CC5303
Sistemas Distribuidos

Sebastián Blasco V.

sblasco@dcc.uchile.cl

CC5303 – Sistemas Distribuidos

7.- Replicación

Parte 1

Sebastián Blasco V.

Contenidos

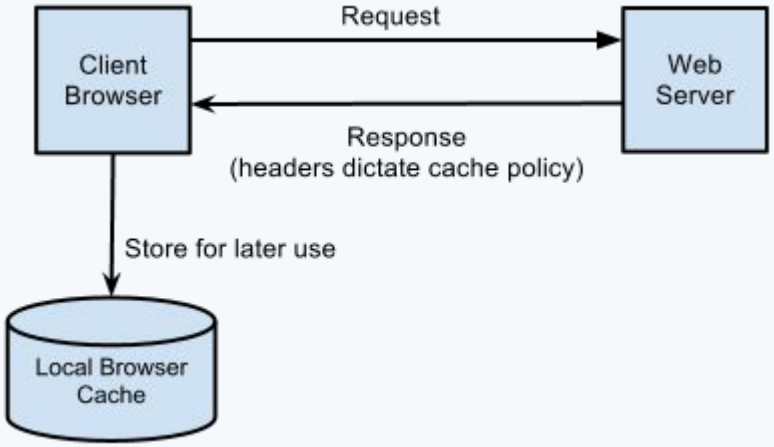
- Replicación
- Modelos de consistencia
  - Centrados en los datos
  - Centrados en el cliente
- Administración de réplicas
- Protocolos de consistencia

Replicación

- Motivos para replicar
  - Confiabilidad
    - Poder seguir trabajando aún cuando se presenten fallas en determinadas réplicas.
      - Caída de un servidor web, continuamos trabajando con un relevo.
    - Tener un mecanismo de protección por datos corruptos.
      - ej. 3 réplicas, se ordena un determinado cálculo, nos quedamos con el resultado con mayoría.
  - Rendimiento
    - Poder escalar en distintas aristas
      - Poder de cómputo, para mejor rendimiento
      - Geográficamente, para mejor disponibilidad
      - Etc.

Replicación

- Mecanismo más simple de replicación
  - Caché



¿Cómo usarlo bien?

Replicación

¿*Trade off* por garantizar consistencia?

- El precio a pagar por replicar los datos es mantener la **consistencia de las réplicas**.
  - Cambiar una réplica la hace diferente del resto de las copias.
  - Actualizar las réplicas consume ancho de banda.
  - Sincronizar las réplicas

- El precio de la consistencia lo determina el **cuándo y cómo** deben realizarse dichas modificaciones.

Replicación

- El precio de la consistencia lo determina el **cuándo y cómo** deben realizarse dichas modificaciones.
- Ej.
  - Considere un proceso *P* que accede a una réplica local *N* veces por segundo, mientras que a la réplica se le actualiza *M* veces por segundo.
  - Suponga que una actualización refresca completamente la versión anterior de la réplica local.
  - Si $N << M$?
    - P nunca accede a muchas versiones actualizadas de la réplica local, ello se traduce en que la comunicación de la red sea inútil para esas versiones.
  - Si $N >> M$?
    - P debería hallar siempre la versión más actual!

Replicación

- La idea principal es que una actualización se realiza en todas las copias como una sola operación atómica, o transacción.
- PERO…
  - Implementar atomicidad involucrando una gran cantidad de réplicas (que pueden estar ampliamente dispersas a través de una red de gran escala) es inherentemente difícil cuando se necesita **completar operaciones rápidamente**.

Replicación

Dilema

- Los problemas de escalabilidad pueden disminuirse aplicando replicación y cacheo, lo que deriva en un **mejor rendimiento**.
- Mantener consistentes a todas la copias generalmente, requiere de una sincronización global, y ello es inherentemente **costoso en términos de rendimiento**.

La cura puede resultar peor que la enfermedad.

Replicación

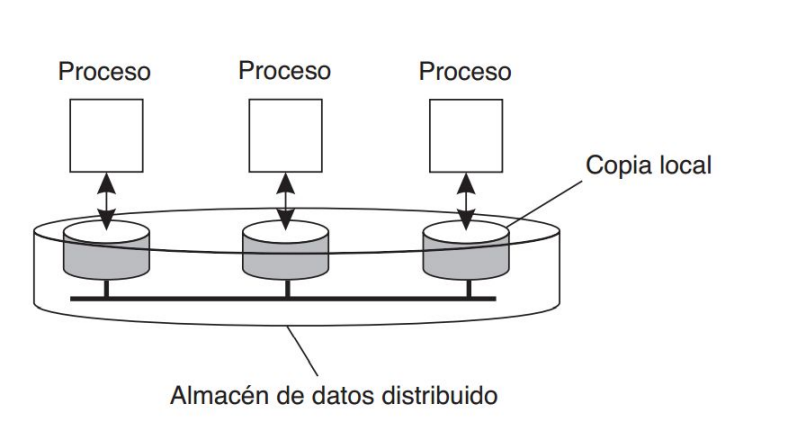
¿Solución?

- Disminuir restricciones de consistencia
  - Relajar el requerimiento de que las actualizaciones necesitan ejecutarse como operaciones atómicas
  - Evitar las sincronizaciones globales (instantáneas)
  - Aumentar así el rendimiento.
- ¿El precio a pagar?
  - **Las copias podrían no ser las mismas en todas partes.**

Hasta dónde relajar la consistencia depende en gran medida de los patrones de acceso y actualización de los datos replicados, así como del propósito de utilizar esos datos.

Modelos de consistencia

- *Def.* Almacén de Datos
  - Es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo.
  - Puede estar físicamente distribuido en varias máquinas.
  - Todo proceso que puede acceder a datos del almacén tiene una copia local disponible de todo el almacén.
  - Las operaciones de escritura se propagan hacia las otras copias.



Modelos de consistencia

- *Def.* Modelo de consistencia
  - Contrato entre los procesos y el almacén de datos que explicita que:

Si los procesos **aceptan** obedecer ciertas reglas, el almacén de datos **promete** funcionar correctamente.

  - Un proceso que realiza una operación de lectura sobre un elemento de datos espera que la operación devuelve un valor que muestre los resultados de la última operación de escritura sobre los datos.

Modelos de consistencia

En la ausencia de un reloj global, es difícil definir precisamente cuál es la última operación de escritura. Como alternativa, necesitamos proporcionar otras definiciones, lo que nos lleva a una gama de modelos de consistencia.

- Centrados en los datos
  - Consistencia Estricta
  - Consistencia Continua
  - Consistencia Secuencial
  - Consistencia Causal
  - Consistencia FIFO
  - Consistencia Débil
  - Consistencia Relajada
- Centrados en el cliente

Consistencia Estricta

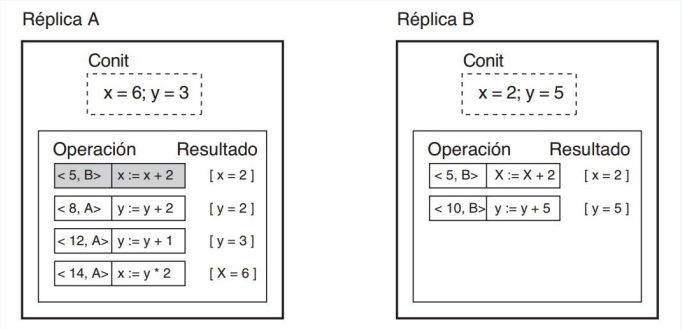
- Es el modelo de consistencia más simple y restrictivo de todos.
- Cualquier lectura sobre un ítem de dato x retorna un valor correspondiente con la más reciente escritura sobre x (en términos de un hipotético reloj de tiempo global)

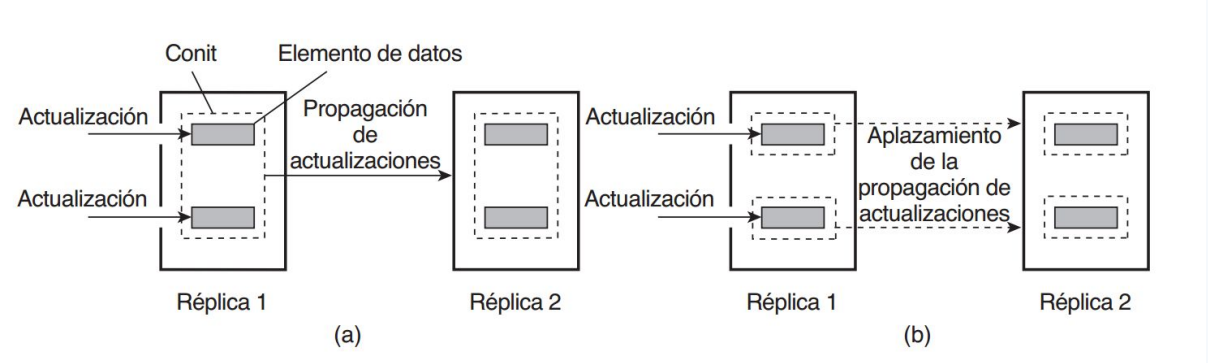
Consistencia Continua

- Hay diferentes formas en que las aplicaciones especifican las inconsistencias que pueden tolerar.
  - Desviación en valores numéricos entre réplicas
    - Ej. la replicación de registros que contienen precio de la UF (+-$100).
  - La desviación numérica también puede comprenderse en términos del número de actualizaciones que se han aplicado a una réplica dada, pero que aún no han sido vistas por otras replicas.
    - Ej. un caché web puede no haber visto un lote de operaciones realizadas por un servidor web.
- Hay clases de aplicaciones en las que se permite que el ordenamiento de actualizaciones sea diferente en varias réplicas, siempre que las diferencias sean limitadas.

Consistencia Continua

- *Def.* Conit
  - Unidad con la que se medirá la consistencia.
  - Se deben definir con cuidado el tamaño de los *conits* (Su granularidad)
    - ¿Grandes/Chicas?
    - ¿Pro/contra?



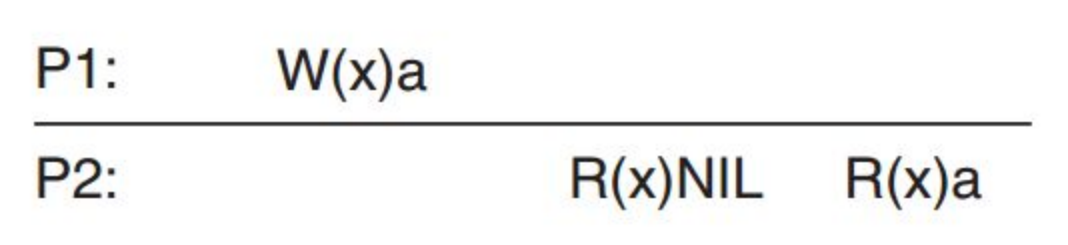


Consistencia Secuencial

- Se dice que un almacén de datos es secuencialmente consistente cuando satisface la siguiente condición:
  - El resultado de cualquier ejecución es el mismo que si las operaciones (de lectura y escritura) de todos los procesos efectuados sobre el almacén de datos se ejecutaran en algún orden secuencial y las operaciones de cada proceso individual aparecieran en esa secuencia en el orden especificado por su programa.
  - **Cualquier interpolación válida de operaciones de lectura y escritura es un comportamiento aceptable, pero todos los procesos ven la misma interpolación de operaciones.**
- Nada se dice sobre el tiempo, no hay referencia a la operación de escritura “más reciente” sobre el elemento de datos

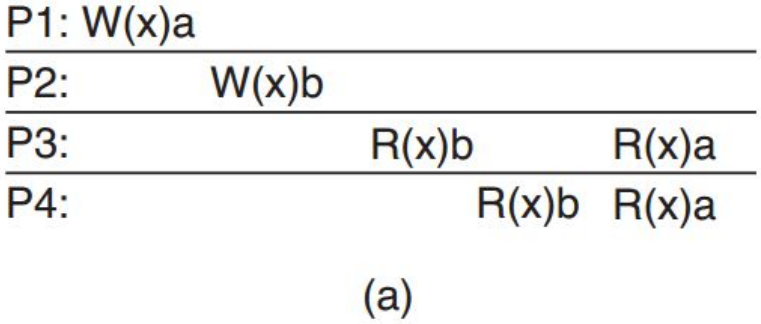
Consistencia Secuencial

- Ej.

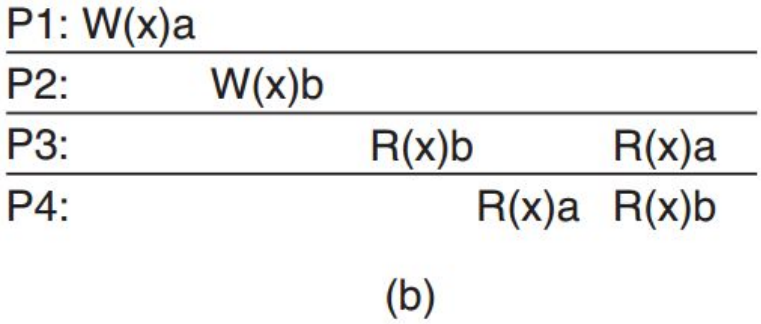


Consistencia Secuencial

- Ej.



(a)



(b)

(a) Almacén de datos secuencialmente consistente.

(b) Almacén de datos que no es secuencialmente consistente.

Las escrituras fueron vistas en otro orden, pero por todos igual

Consistencia Secuencial

- Ej.

Proceso 1	Proceso 2	Proceso 3
x ← 1;	y ← 1;	z ← 1;
impresión (y, z)	impresión (x, z)	impresión (x, y)

- Total: 6 instrucciones
- Posibles permutaciones: 6! => 720 posibles ordenamientos.
- Si se fija la primera operación (x ← 1): 5! => 120 pos.
- De las 120 posibilidades… ¿Son todas válidas?
  - Sólo son válidas las que respeten secuencialidad de las op. de los procesos! => ¼ del total: 30
- Dadas las 3 posibles primeras operaciones: 30*3 = 90 posibilidades de ordenamientos de ejecución.

Consistencia Secuencial

- Ej.

Proceso 1	Proceso 2	Proceso 3
x ← 1;	y ← 1;	z ← 1;
impresión (y, z)	impresión (x, z)	impresión (x, y)

x ← 1;	x ← 1;	y ← 1;	y ← 1;
impresión (y, z);	y ← 1;	z ← 1;	x ← 1;
y ← 1;	impresión (x, z);	impresión (x, y);	z ← 1;
impresión (x, z);	impresión (y, z);	impresión (x, z);	impresión (x, z);
z ← 1;	z ← 1;	x ← 1;	impresión (y, z);
impresión (x, y)	impresión (x, y)	impresión (y, z)	impresión (x, y);
Firma: 001011	Firma: 101011	Firma: 010111	Firma: 111111
(a)	(b)	(c)	(d)

Consistencia Causal

- Representa una suavidad de las restricciones de la consistencia secuencial, ya que **sólo diferencia entre eventos que potencialmente están relacionados por la causalidad** y los que no lo están.

- Causalidad?
  - Recuerdo: Si el evento b es causado o influenciado por un evento previo a, la causalidad requiere que todos los demás eventos vean primero a a, y después a b.

Consistencia Causal

- Causalidad
  - Considere una simple interacción mediante una base de datos distribuida compartida.
  - Suponga que el proceso *P1* escribe un elemento de datos *x*. Después *P2* lee a *x* y escribe *y*.
  - Aquí, la lectura de *x* y la escritura de *y* están relacionados potencialmente por la causalidad, ya que el cálculo de *y* pudo haber dependido del valor de *x* cuando *P2* lo leyó (es decir, el valor escrito por *P1*).
  - Si dos procesos escriben espontánea y simultáneamente dos diferentes elementos de datos, éstos no están causalmente relacionados. Se dice que las operaciones que no están causalmente relacionadas son concurrentes.

Consistencia Causal

Para que a un almacén de datos se le considere causalmente consistente, es necesario que obedezca la siguiente condición:

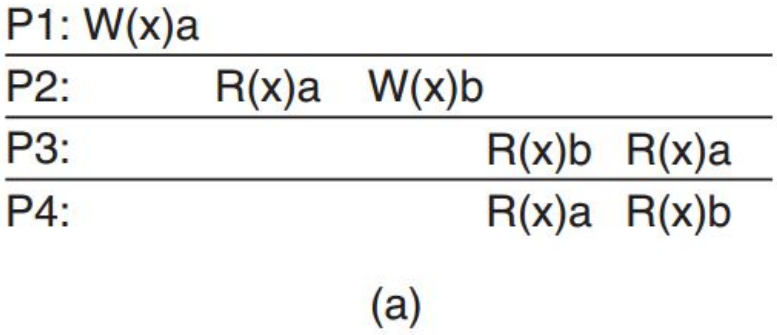
- Escrituras que potencialmente están relacionadas por la causalidad, **deben ser vistas por todos los procesos en el mismo orden**. Las escrituras concurrentes pueden verse en un

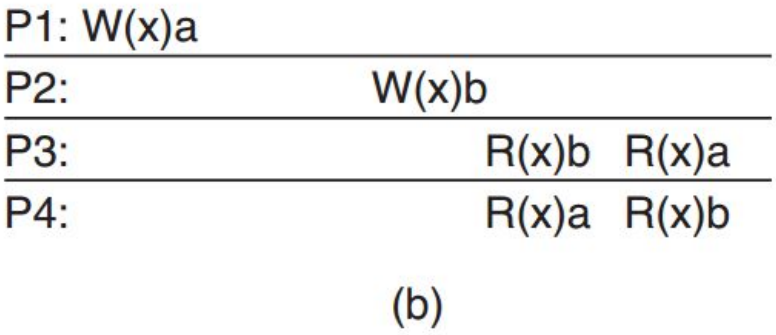
P1: W(x)a			W(x)c		
P2:	R(x)a	W(x)b			
P3:	R(x)a			R(x)c	R(x)b
P4:	R(x)a			R(x)b	R(x)c

Esta secuencia está permitida con un almacén causalmente consistente, pero no con un almacén secuencialmente consistente.

Consistencia Causal

- Ej.



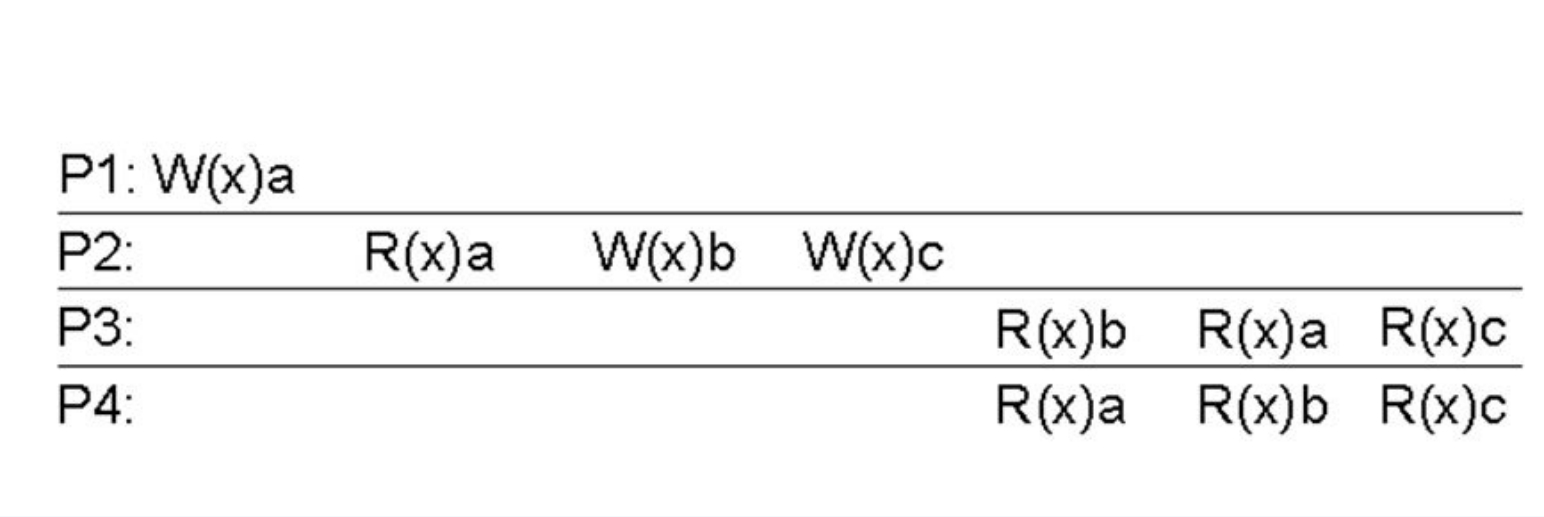


(a) Violación a un almacén causalmente consistente.

(b) Secuencia de eventos correcta en un almacén causalmente consistente.

Consistencia FIFO

- Toda escritura realizada por un proceso debe ser vista por los otros procesos **en el orden que fueron realizadas**.
- Escrituras desde diferentes procesos pueden ser vistas en orden distinto por diferentes procesos.



Consistencia FIFO

- Ej.

Process P1	Process P2
x = 1;	y = 1;
if (y == 0) kill (P2);	if (x == 0) kill (P1);

- ¿Qué procesos mueren?
  - Intuitivamente: P1, o P2, o ninguno.
  - Con consistencia FIFO pueden morir ambos:
    - P1 hace R1(y)0 antes de ver el W2(y)1
    - P2 hace R2(x)0 antes de ver el W1(x)1
  - ¿Por qué? Porque sólo las escrituras de un mismo proceso se ven en orden

Consistencia Débil

- Este esquema es muy restrictivo: no todos los procesos quieren ver todas las escrituras (puede que no estén relacionados).
- Lo mejor es una vez que se tenga un resultado final, propagarlo.
  - ¿Cómo saber cuando se tiene un resultado final?

- Secciones críticas
  - Variables de sincronización (S)

Consistencia Débil

Se deben cumplir los siguientes criterios (Bershad y cols., 1993)

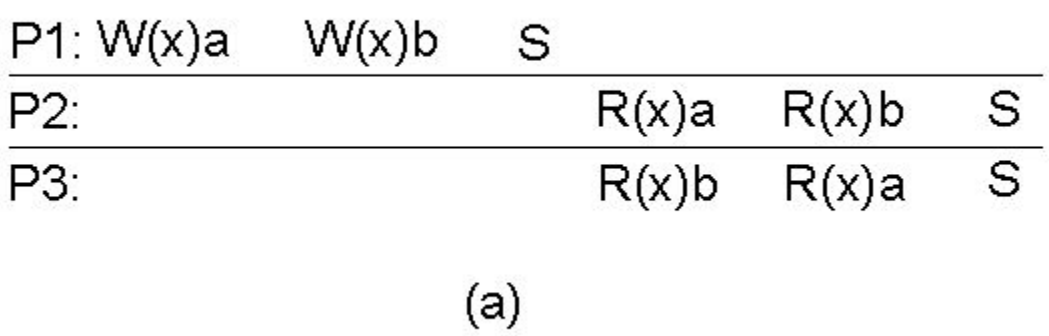
1. El acceso para adquirir una variable de sincronización con respecto a un proceso no está permitido sino hasta que se realizan todas las actualizaciones de los datos compartidos con respecto a ese proceso.
2. Antes de que a un proceso se le permita un modo exclusivo de acceso a una variable de sincronización, ningún otro proceso puede tener a la variable de sincronización, ni siquiera en modo no exclusivo.
3. Después de que se ha realizado un acceso en modo exclusivo hacia una variable de sincronización, ningún otro acceso de modo no exclusivo de otro proceso hacia esa variable de sincronización puede realizarse, sino hasta que se haya realizado con respecto al propietario de esa variable.

Consistencia Débil

- La sincronización fuerza a que todos los procesos vean la última actualización (escritura) sobre una variable.
- La consistencia débil asegura consistencia sobre un grupo de operaciones, no sobre lecturas o escrituras aisladas.
- Útil, por ejemplo, en transacciones distribuidas.

Consistencia Débil

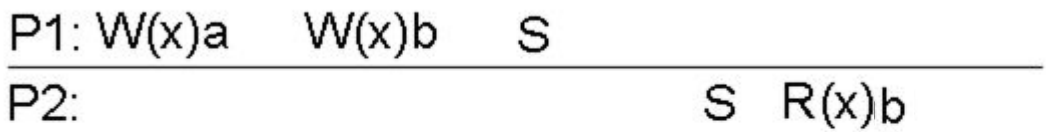
- Ej.



(a)

Como P2 y P3 no han sincronizado, pueden ver las escrituras en cualquier orden

Al sincronizar, sólo hay un resultado consistente



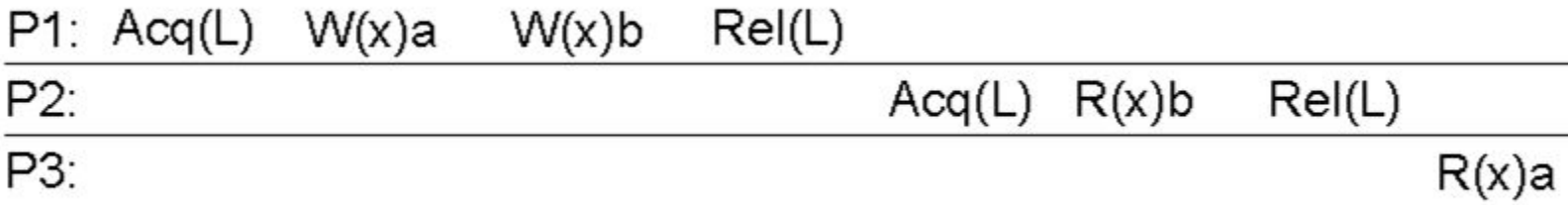
(b)

Consistencia relajada

Se utilizan dos operaciones:

- Acquire: avisa la entrada a la sección crítica
- Release: avisa la salida de la sección crítica

- También reemplazable por barreras: ningún proceso puede seguir hasta que todos hayan alcanzado el punto de sincronización.



P3 no hizo uso de la variable de sincronización, por lo tanto puede leer cualquiera de sus valores en el tiempo

Consistencia relajada

La consistencia relajada garantiza que:

- Al hacer un acquire, las copias de los datos protegidos estarán actualizadas.
- Al hacer un release, las actualizaciones se propagarán a las otras copias.

Pero:

- Hacer un acquire no garantiza que los cambios locales serán propagados inmediatamente.
- Hacer un release no garantiza que se actualizarán datos desde otras copias inmediatamente.

Resumen Modelos de Consistencias

Consistency	Description
Estricta	Ordenamiento en tiempo absoluto de todos los accesos compartidos.
Linealizada	Todos los procesos deben ver todos todos los accesos compartidos en el mismo orden. Los accesos son ordenados de acuerdo a una marca de tiempo global.
Secuencial	Todos los procesos ven todos los accesos compartidos en el mismo orden. Los accesos no están ordenados en el tiempo.
Causal	Todos los procesos ven los accesos compartidos causalmente relacionados en el mismo orden.
FIFO	Todos los procesos ven las escrituras de un proceso en el orden que éste las efectuó. Escrituras de diferentes procesos pueden no ser vistas en el mismo orden.

(a) Modelos de consistencia que no usan variables de sincronización

Consistency	Description
Débil	Los datos compartidos pueden ser considerados consistentes luego de que se haya hecho la sincronización.
relajada	Los datos compartidos son hechos consistentes cuando la región crítica es abandonada.
Entry	Los datos compartidos pertenecientes a una región crítica son hechos consistentes cuando se entra a la región crítica.

(b) Modelos de consistencia con operaciones de sincronización