Dpto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

ELEMENTOS DE BASES DE DATOS

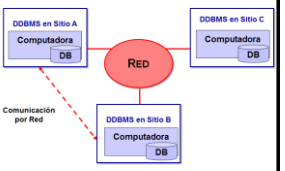
Segundo Cuatrimestre 2013

Clase 21: SMBDD – Bloqueo de Datos Lógico

Mg. María Mercedes Vitturini
[mvitturi@cs.uns.edu.ar]

Bases de datos distribuidas

- Un sistema de bases de datos distribuido (DDBMS ó SMBDD) se compone de varios **sitios o nodos** levemente acoplados que no comparten componentes físicos.
- Los nodos se conectan a través de la red.
- Los nodos pueden variar en tamaño y función.
- Generalmente se ubican geográficamente separados.
- Cada DBMS de un sitio funciona independientemente de los otros.



Bases de Datos Distribuidas

- Bases de datos distribuidas **homogéneas (SMBDD)**:
  - El mismo software/esquema de base de datos en los sitios.
  - Los datos (instancias) están particionados y/o replicados en los sitios.
  - Objetivo: proveer los usuarios la vista de una sola base de datos ocultando detalles de distribución.
- Bases de datos distribuidas **heterogéneas**:
  - Diferentes software/esquema en los sitios.
  - Objetivo: integrar distintas bases de datos para proveer una función útil.

SMBDD: gestión de datos

Replicación – el sistema mantiene varias copias (réplicas) idénticas de una relación *r*. Cada réplica se almacena en un sitio diferente. Mejoras

- Tiempo de respuesta y tolerancia a fallos.

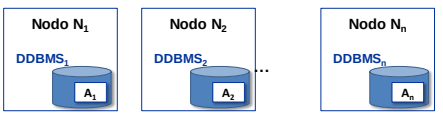
Fragmentación – una relación *r* se divide en varios fragmentos almacenados en sitios diferentes.

Replicación y Fragmentación – una relación *r* es particionada en varios fragmentos y el sistema mantiene varias copias idénticas de éstos fragmentos.

Datos Lógicos y Datos Físicos

- En DDBMS ó SMBDD se identifican los conceptos "*dato*"

DATO LÓGICO Y **DATO FÍSICO**



A es el dato lógico **A_i es el dato físico**

- Si de un dato existe una única copia ⇒ dato lógico ≡ dato físico.
- En entornos distribuidos homogéneos con replicación la gestión de bloqueos debe realizarse a nivel de dato lógico.
- El criterio de compatibilidad entre bloqueos es la misma.

Gestor de Bloqueos Distribuido

- Entre los cambios a incorporar en el SMDBD está *cómo debe operar el Gestor de Bloqueos del SMDBD si cuando existen múltiples copias de los datos*, de forma tal que:
  - *Múltiples transacciones* puedan obtener un **lock compartido sobre A**, pero *una única transacción* por vez puede obtener un **lock exclusivo sobre A**.
  - Además, en caso de actualización, asegurar hay que *todas las copias de un ítem de datos* se pueden **actualizar** y así permanecer iguales.

Gestión de Bloques en DDBMS

- Algunas alternativas para implementar la traducción entre "lock de dato lógico" en "locks de datos físicos " en entornos distribuidos con réplicas:
  - **Utilizar un Gestor de Bloqueos Centralizado**.
    - Nodo Central
  - **Utilizar Gestión de Bloqueos Distribuida.**
    - ROWA
    - Mayoría
    - K de n
  - **Otras propuestas híbridas**
    - Sitio primario (o copia primaria)
    - Token primario

Gestión de Bloques en DDBMS

- La comunicación entre los sitios o nodos se realiza por medio de *mensajes*.
- La cantidad de mensajes a transmitir "condiciona" la performance general del sistema. Se distinguen:
  - **Mensajes de Control**: que indican requerimientos o concesión de bloqueos, estados de cometido o de aborto, estados de deadlock, etc.
  - **Mensajes de Datos:** que contienen datos puros, leídos de o a ser escritos en la base de datos.
- Bajo ciertas condiciones los mensajes de control cuestan lo mismo que los de datos, pero en general los mensajes de datos son más caros.

Gestor de Bloqueos Centralizado: Nodo Central

- El DDBMS define como *gestor de bloqueos a un sitio S.*
- Si una transacción en un sitio S_i necesita bloquear un ítem de dato Q, envía el requerimiento a *S,* que determina si se puede conceder:
  - Si es compatible, *S* envía el mensaje al sitio S_i que lo solicitó.
  - Sino, el S_i quedará demorado hasta que se le pueda asignar el bloqueo.

Ventajas ☺ :

- Implementación simple y control de deadlock distribuido simple.

Desventajas ☹:

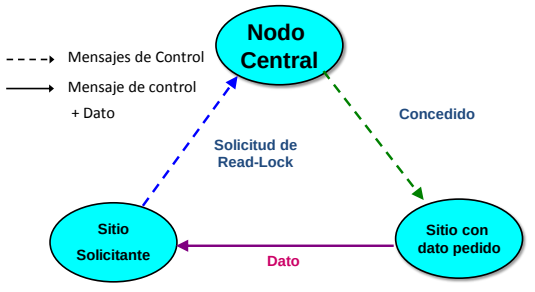
- *S* es el "lock manager" se transforma en cuello de botella y el DDBMS es vulnerable al fallo del sitio *S.*

Mensajes con Nodo Central

Para obtener un **read-lock:**

- S_i envia un mensaje requiriendo un *read-lock* al nodo central *S*.
- Si el *bloqueo no es concedido*, S_i se queda esperando.
- Si el *bloqueo es concedido*, el nodo central *S* envía un mensaje a un sitio S_k que tiene una copia del ítem de dato.
- Luego, el sitio S_k envía un mensaje con el valor del dato al sitio que hizo el requerimiento.

Lock-S con Nodo Central



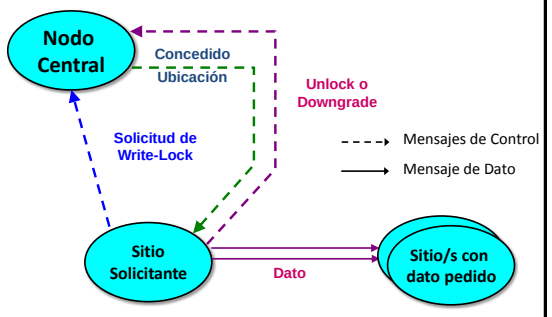
Método de Nodo Central

Para obtener un **write-lock:**

- El sitio S_i que necesita actualizar un dato envía un mensaje requiriendo un *write-lock* al nodo central.
- Si *el bloqueo no es concedido*, el sitio S_i se queda esperando.
- Si *el bloqueo es concedido*, el nodo central le contesta con un mensaje notificando el/los sitio/s que tiene/n copia del ítem de dato. Luego, el/los sitio/s con la copia del ítem de dato reciben el nuevo valor a escribir.
- S_i envia un mensaje adicional liberando el bloqueo de escritura una vez que el dato fue actualizado.

Estas transparencias proveen sólo una referencia a los temas. Para su estudio debe remitirse a la bibliografía.

Lock-X con Nodo Central



Método de Nodo Central

Análisis del método Nodo Central

- ☹ Se requieren de mensajes de control extra: al nodo central y al nodo donde reside la copia del dato.
- ☹ El nodo central es el cuello de botella de la performance de la red, la mayoría de los mensajes vienen de él o van hacia él.
- ☹ La falla del sitio que opera de nodo central torna inoperativo al sistema.
- ☺ Es sencillo de implementar y de detectar y manejar situaciones de *deadlock*.

Gestión de Bloqueos Híbrida: Sitio primario

- Define un sitio primario o "nodo copia primaria" para cada ítem de dato.
- Así, la responsabilidad en la administración de los bloqueos para un ítem lógico *A* se deja a cargo de un sitio particular, sin importar cuántas copias del mismo dato existan.
  - Ejemplo: sea una base de datos de un banco y los nodos de la red sucursales del banco, es natural considerar el sitio primario del ítem cuenta a la sucursal que creó y tiene esa cuenta.
- Si el sitio primario *S* del ítem *A* no es el nodo S_i donde se ejecuta la transacción, entonces S_i envía un requerimiento de bloqueo al gestor de *S* y se espera por su respuesta.

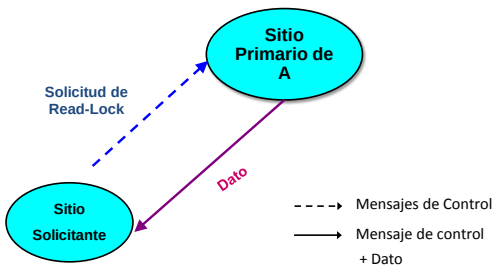
Sitio primario

Varios sitios comparten la responsabilidad de administrar bloqueos:

- La gestión de bloqueos distribuida se implementa colaborando entre varios gestores de bloqueos ubicados en distintos nodos.
- Cada gestor de bloqueos controla el acceso a datos de los que tiene una copia local.
- Se define un protocolos para actualizar todas las copias.

- ☺ Ventajas:
  - Distribuye el trabajo,
  - El sistema es más robusto ante fallos.

Lock-S con Sitio Primario



Gestión de bloqueos distribuida: Read-Locks-One / Write-Locks-All

- Para obtener un **read-lock** (*lock-s*) sobre un ítem de dato A, la transacción debe obtener un lock-S sobre *una de las copias* de A.
- Para obtener un **write-lock** (*lock-x*) sobre un ítem de dato A, la transacción debe obtener el lock sobre *todas las copias* de A.

Las reglas para conceder bloqueos son:

- Un *read-lock* se consigue sobre una copia si ninguna otra transacción tiene un write-lock sobre la copia.
- Un *write-lock* es concedido sobre una copia si ninguna otra transacción tiene un read-lock o un write-lock sobre la copia.

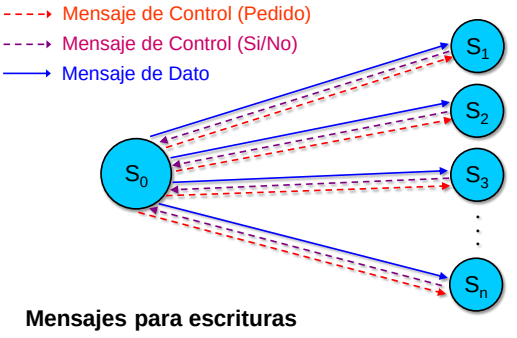
Estas transparencias proveen sólo una referencia a los temas. Para su estudio debe remitirse a la bibliografía.

ROWA - Análisis

Si existen ***n* copias de un dato A.**

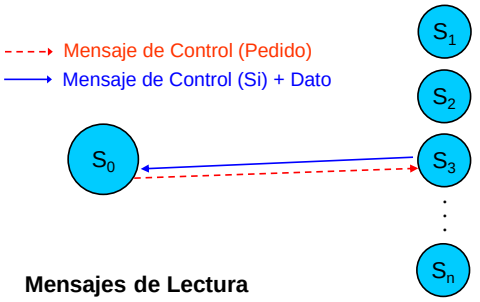
- Para obtener **read-lock sobre A**, se necesita conocer un sitio donde exista una copia de A para enviarle el requerimiento de bloqueo. Si se concede el bloqueo: 1 mensaje de control y 1 mensaje de datos. La concesión del bloqueo viene con el dato.
- Para obtener **write-lock sobre A**, se necesitan enviar *2n mensajes de control* (*n* para requerir el bloqueo y *n* de concesión del mismo) y *n* mensajes de datos.

ROWA: lock-X



Mensajes para escrituras

ROWA: lock-s

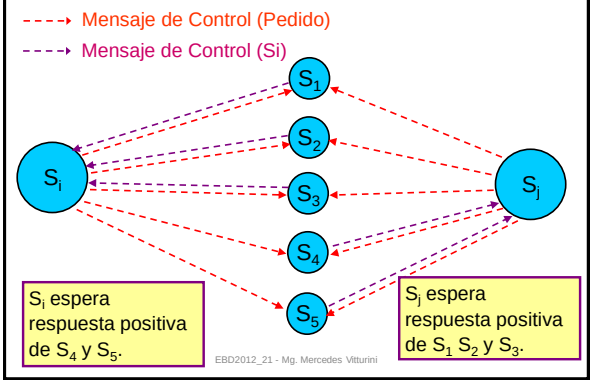


Mensajes de Lectura

ROWA

- ROWA se comporta apropiadamente en entornos donde predominan las lecturas.
  - Ejemplo: una guía telefónica on-line, donde los datos se actualizan en períodos fijos y las lecturas son en todo momento las operaciones predominantes.
- Si se solicita más de un pedido de escritura (lock-X) simultáneamente y el mismo no puede ser otorgado, si se decide esperar se puede entrar en *deadlock*.
- Cuando se solicita un pedido de lectura y el mismo no puede ser otorgado, en general se espera (lo más probable es que otro sitio esté escribiendo el dato).

Deadlocks en escrituras en ROWA



Majority Locking

- Para obtener un **read-lock sobre A**, una transacción debe obtener un read-lock sobre *la mayoría de las copias* de A.
- Para obtener un **write-lock sobre A**, una transacción debe obtener un write-lock sobre *la mayoría de las copias* de A.

Las reglas para conceder bloqueos son iguales al protocolo anterior. Si se asume que el número (*n*) de copias la mayoría es el techo de (*n*+1)/2.

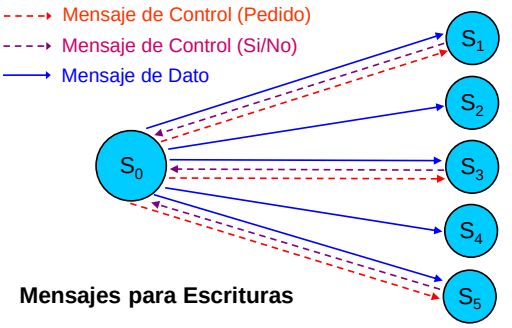
Ejemplo: la mayoría para *n* = 5 es 3, y la mayoría para *n* = 6 es 4.

Majority Locking - Análisis

Si existen *n* copias de un dato A.

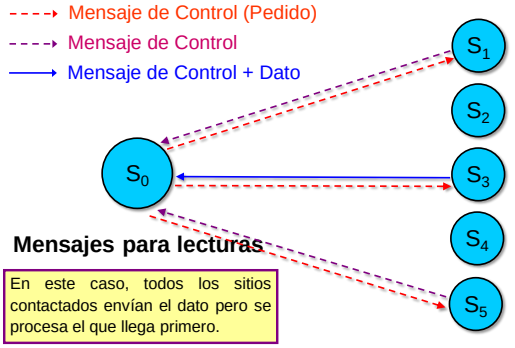
- Para realizar un *write-lock A*, se envian *n+1* mensajes de control ((*n*+1)/2 para requerir el bloqueo y (*n*+1)/2 de concesión del mismo) y *n* mensajes de datos (se escriben todas las copias).
- El número de mensajes de control es similar si se realiza un *read-lock* pero se atiende sólo 1 mensaje de datos.
- Si la transacción corre en el sitio de una de las copias, se pueden omitir algunos mensajes (no se transfieren por la red).

Majority Locking: lock-x



Mensajes para Escrituras

Majority Locking: lock-s



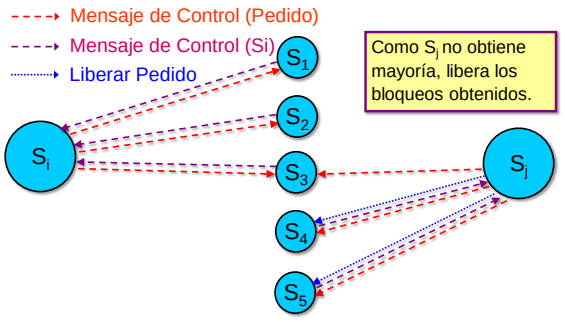
Mensajes para lecturas

En este caso, todos los sitios contactados envían el dato pero se procesa el que llega primero.

Majority Locking

- Este protocolo se comporta de manera apropiada si predominan las escrituras.
  - –Ejemplo: un sistema de reserva de pasajes, donde los datos se modifican constantemente.
- Cuando se solicitan simultáneamente pedidos, sólo un sitio puede obtener la mayoría por lo que existe menor riesgo de deadlock.
- Si se solicita un dato y fracasa el pedido, se puede esperar que se liberen copias para obtener la mayoría, o bien se aborta la transacción.
- En caso de espera, debe garantizarse que un sitio no entre en estado de inanición (en espera continua).

Menor riesgo de deadlocks en escrituras



Como S_i no obtiene mayoría, libera los bloqueos obtenidos.

Estrategia *k*-de-*n*

Si existen *n* copias del dato. Sea *k* un valor tal que $n/2 < k \leq n$.

- Para obtener un **write-lock A**, una transacción debe obtener un write-lock sobre *k copias* de A.
- Para obtener un **read-lock A**, una transacción debe obtener un read-lock sobre un total de *n-k+1 copias* de A.

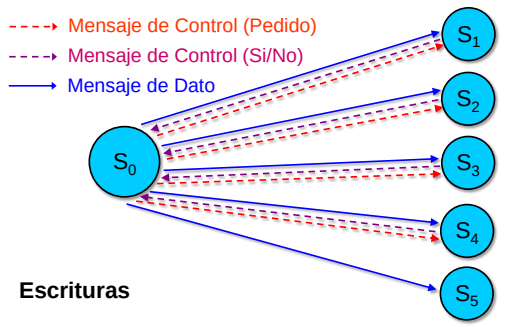
✓ No es posible que existan read-locks y write-locks simultáneamente (se necesitarían *n*+1 copias de A).
✓ Tampoco pueden existir dos write-locks simultáneamente ($k > n/2$).

Estas transparencias proveen sólo una referencia a los temas. Para su estudio debe remitirse a la bibliografía.

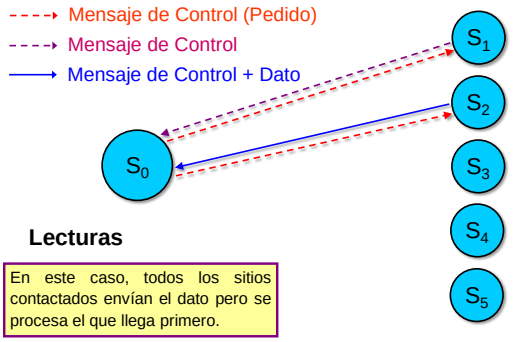
Estrategia k-de-n

- La estrategia *n-de-n* deriva en Write-Locks-All.
- La estrategia *(n+1)/2-de-n* deriva en Majority Locking.
- A medida que *k* se incrementa, el protocolo se desempeña mejor en situaciones en donde se realizan lecturas más frecuentemente.
- A medida que *k* se decrementa, el protocolo se desempeña mejor en situaciones donde predominan las escrituras.

Estrategia 4-de-5: lock-x



Estrategia 4-de-5: lock-s



En este caso, todos los sitios contactados envían el dato pero se procesa el que llega primero.

Tokens de Copia Primaria

- Este método asume la existencia de **read-tokens** y **write-tokens**, o privilegios que los nodos de la red pueden obtener, en beneficio de sus transacciones y con el fin de acceder a los ítems.
- Para un ítem *A*, puede existir sólo un *write-token*. Si no existe un write-token, puede existir cualquier número de *read-tokens*.
  - Si un sitio tiene un ***write-token*** para *A*, entonces puede conceder un *read-lock* o un *write-lock* para *A*.
  - Si un sitio tiene un ***read-token*** para *A*, entonces solamente puede conceder un *read-lock* para *A*.

Tokens de Copia Primaria

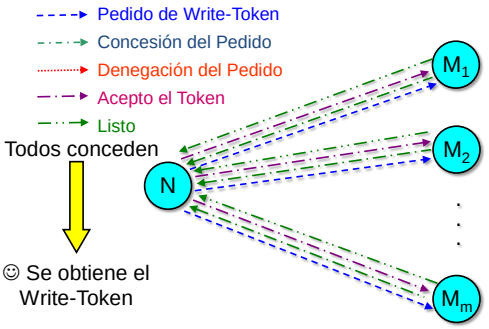
- Si una transacción en un sitio *N* desea un *write-lock* para *A*, debe obtenerlo para el sitio.
  - Si el *write-token* está en ese sitio entonces no hace nada.
  - Si el *write-token* no está en ese sitio entonces

Continua ...

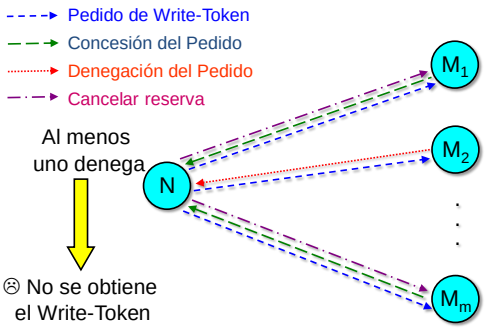
El *write-token* no está en el sitio entonces ...

- *N* envía un mensaje a todos los sitios requiriendo el *write-token.*
- Cada sitio *M* que recibe el requerimiento y contesta:
  (a) *M* no tiene un *read/write-token* para *A* o está por liberarlo de modo que *N* puede obtener el write-token.
  (b) *M* tiene un *read/write-token* para A y no lo liberará (otra transacción lo está usando o ha sido reservado a otro sitio).
- Si todos los sitios contestan (a), *N* puede obtener el *write-token* y envía un mensaje a cada sitio diciendo que aceptó el *write-token* y que deberían destruir el que ellos tienen.
- Si algunos contestan (b), *N* no puede obtener el *write-token* y debe enviar un mensaje a los sitios que contestaron (a) para que cancelen la reserva sobre *A.*

Token de Copia Primaria



Token de Copia Primaria



Comparación de métodos

Método	Mensajes de control en escritura	Mensajes de control en lecturas	Observaciones
ROWA	2n	1	Bueno en ambientes donde predominan lectura
Mayoría	>= n+1	>=n	Compensa mensajes en lecturas y escrituras
Nodo Central	3	2	Simple y vulnerable a fallos
Sitio primario	2	1	Eficiente y vulnerable a fallos
Token primario	0-4m	0-4m	Se adapta a cambios temporarios

***n*: número de copias del dato** ***m*: número de nodos en la red**

Temas de la clase de hoy

- Sistemas Distribuidos
  - Protocolos de bloqueo con replicacion
- **Bibliografía**
  - "Principles of Database and Knowledge-Base Systems" – J. Ullman. Capítulo 10.
  - "Database Systems Concepts" – A. Silberschatz. Capítulo 22 (ed. 2005). Capítulo 19 (ed. 2010)

Estas transparencias proveen sólo una referencia a los temas. Para su estudio debe remitirse a la bibliografía.