Bases de Datos Distribuidas

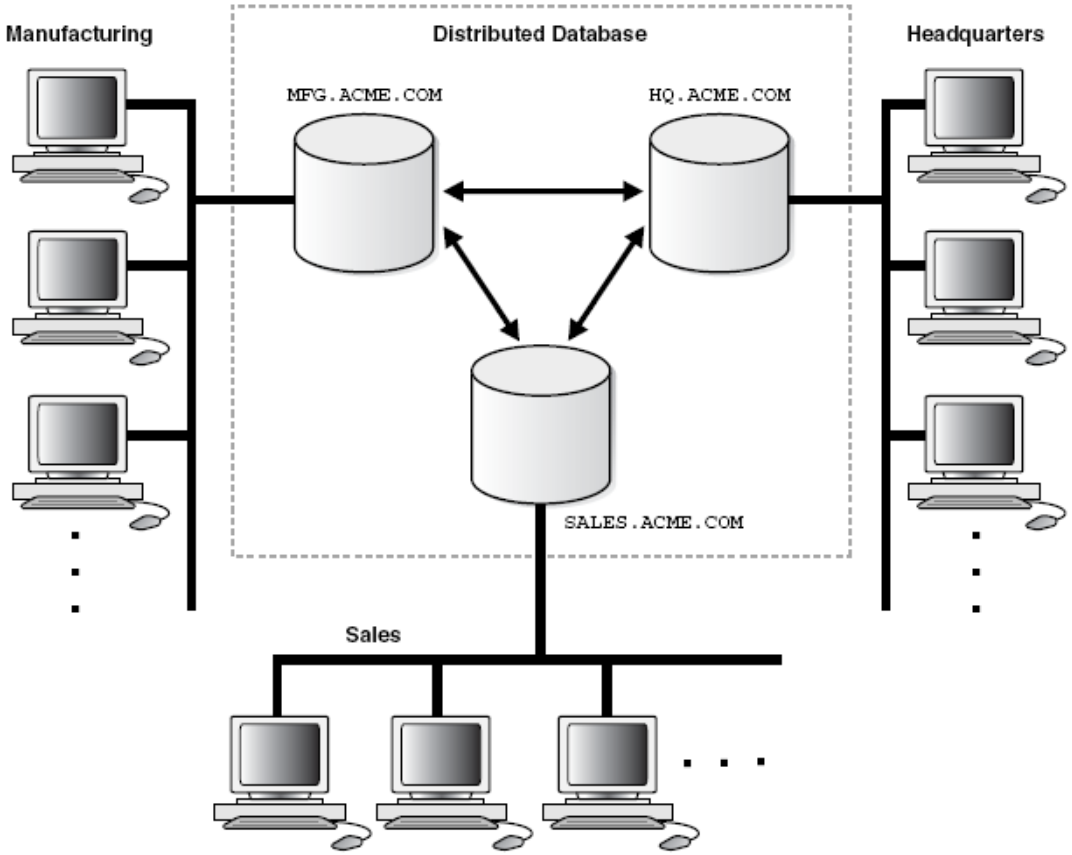
Ing. Joaquín Lima

Bases de Datos II

Facultad Politécnica - UNA

Sistemas de Bases de Datos Distribuidas

- Un Sistema de Bases de Datos Distribuidas (SBDD) es un conjunto de sitios (servidores) débilmente acoplados y que no comparten componentes físicos.
- Cada sitio corre un SGBD que es independiente de los demás.
- Las Transacciones en un SBDD pueden acceder a datos de uno o más sitios.



Tipos de BDD

- Según el Software SGBD utilizado para la implementación de una BDD se tienen dos tipos de BDDs.

- BDD Homogéneas:
  - Todos los sitios utilizan el mismo Sw SGBD.
  - Todos los sitios están comprometidos unos con otros en cooperar para procesar las peticiones de usuario
  - Cada sitio sacrifica parte de su autonomía en términos de modificación del esquema o del Sw SGBD
  - Aparecen frente al usuario como un sistema único.

- BDD Heterogéneas
  - Los sitios pueden usar diferentes SW y Esquemas
    - La diferencia de esquemas es el problema principal en el procesamiento de consultas
    - La diferencia de Sw es el problema principal en el procesamiento de transacciones
  - Los sitios por lo general solo ofrecen características limitadas para el procesamiento de las transacciones en cooperación.

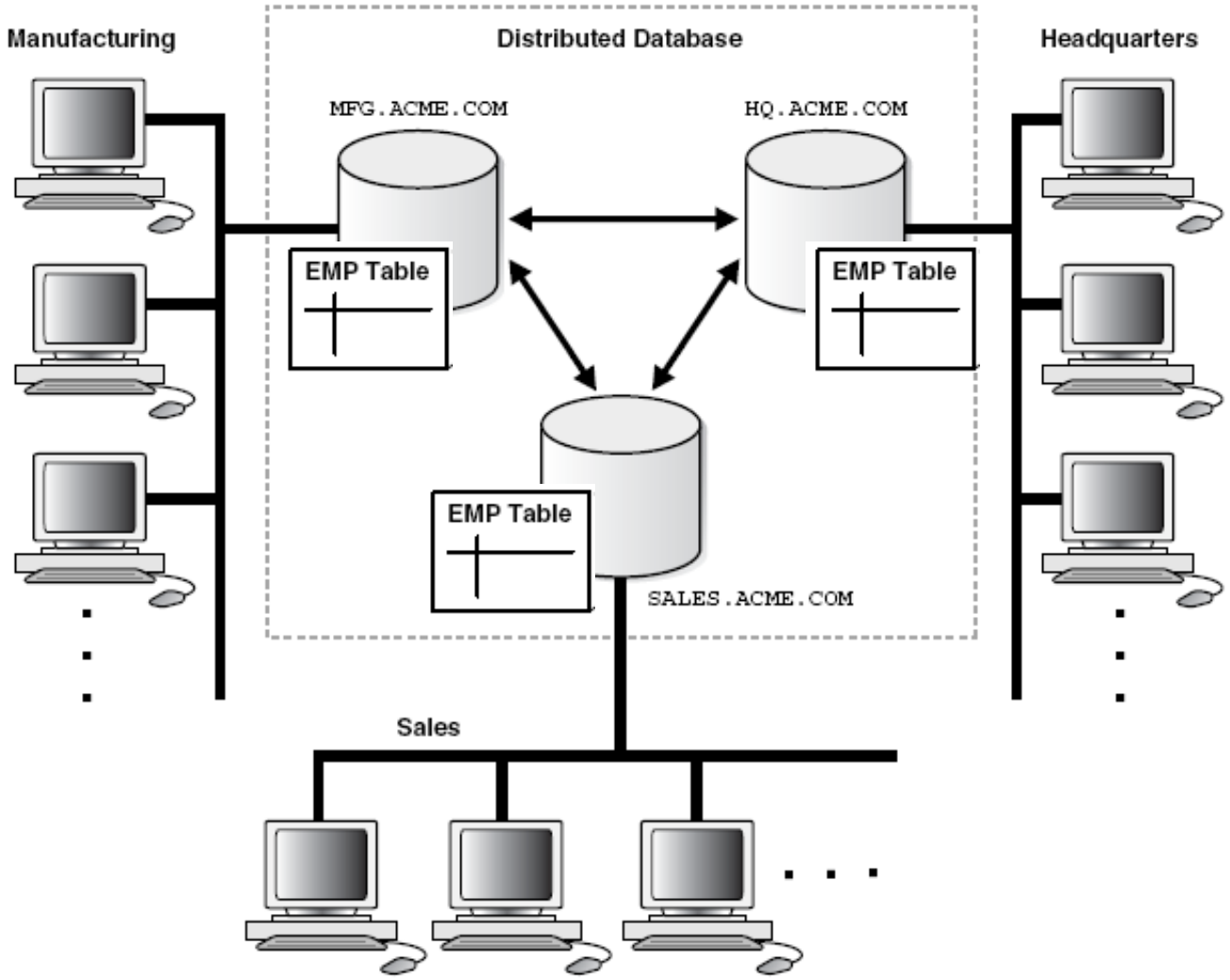
Almacenamiento Distribuido

- Asumiendo un modelo de datos relacional, los tipo de almacenamiento distribuido de datos son
  - Replicación
    - El SBDD mantiene múltiples copias de la misma relación en diferentes sitios.
  - Fragmentación
    - EL SBDD permite partir una relación en varios fragmentos y ubicarlos e varios sitios

Replicación de Datos

- Consiste en guardar copias de una relación o un fragmento en uno o más sitios
- La **Replicación Completa** consiste en almacenar una copia de la relación en cada sitio
- Una **BDD Completamente Redundante** es aquella en que cada sitio contiene una copia entera de la BD.
- Ventajas
  - **Disponibilidad:** La falla del sitio donde se ubica la relación **R** no resulta en la indisponibilidad de los datos de **R.**
  - **Paralelismo:** múltiples consultas de lectura sobre **R** pueden ser procesadas en paralelo por varios sitios.
  - **Trasferencia de Datos Reducida:** Solo las actualizaciones en **R** provocan que varios datos sean transferidos por la red.
- Desventajas
  - **Costo de Actualización Incrementado:** cada replica de **R** debe ser actualizada, lo cual puede provocar un alto trafico en la red.
  - **Complejidad de Control de Concurrencia Aumentada:** la actualización concurrente de las replicas requiere implementar mecanismos especiales de control de concurrencia.
    - **Solución: Utilizar Copias Primarias.**

Replicación de Datos

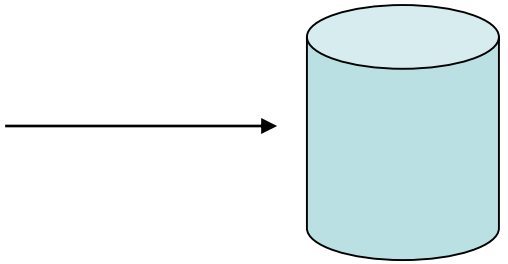


Fragmentación de Datos

- División de la relación R en fragmentos R1, R2, …, Rn, los cuales contienen informacion suficiente para recostruir R.
- **Fragmentación Horizontal:**
  - Cada fila de R es asignado a uno o mas fragmentos
  - $Ri = \sigma_{< condicion\ i >} (R)$
  - R = R1 U R2 U … U Rn
- **Fragmentación Vertical**
  - El esquema de la relación R es partido en dos o más subesquemas.
    - Todo los subesquemas deben contar con una clave candidata que permita la reunión natural de los fragmentos
    - Se puede usar el **id de la tuplas** como clave de reunión, de esta manera todos los subesquemas deben contener este atributo.
  - $Ri = \pi_{< subesquema\ i >} (R)$
  - R = R1 join R2 join … join Rn

Fragmentación Horizontal

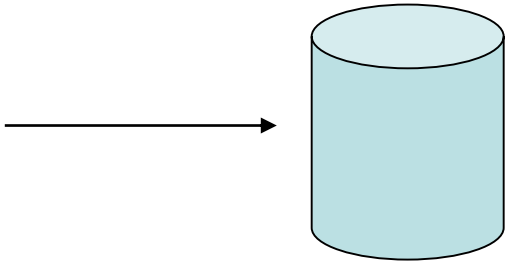
branch_name	*account_number*	*balance*
Hillside	A-305	500
Hillside	A-226	336
Hillside	A-155	62



Suc. Hillside

$account_1 = \sigma_{branch\_name=\text{“Hillside”}}(account)$

branch_name	*account_number*	*balance*
Valleyview	A-177	205
Valleyview	A-402	10000
Valleyview	A-408	1123
Valleyview	A-639	750

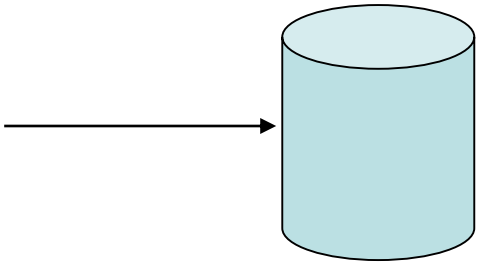


Suc. Valleyview

$account_2 = \sigma_{branch\_name=\text{“Valleyview”}}(account)$

Fragmentación vertical

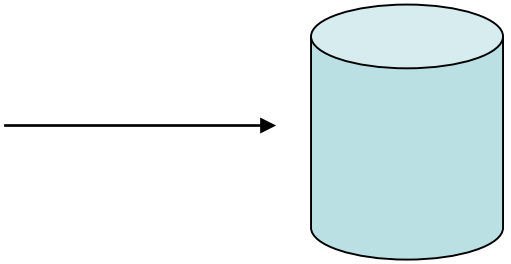
branch_name	*customer_name*	*tuple_id*
Hillside	Lowman	1
Hillside	Camp	2
Valleyview	Camp	3
Valleyview	Kahn	4
Hillside	Kahn	5
Valleyview	Kahn	6
Valleyview	Green	7



Marketing

$deposit_1 = \Pi_{branch\_name,\ customer\_name,\ tuple\_id}(employee\_info)$

account_number	*balance*	*tuple_id*
A-305	500	1
A-226	336	2
A-177	205	3
A-402	10000	4
A-155	62	5
A-408	1123	6
A-639	750	7



Contabilidad

$deposit_2 = \Pi_{account\_number,\ balance,\ tuple\_id}(employee\_info)$

Ventajas de la Fragmentación

- Horizontal
  - Permite el procesamiento en paralelo de los fragmentos
  - Permite que las tuplas se ubiquen el sitio donde son mas frecuentemente accedidas.
- Vertical
  - Permite que los fragmentos sean almacenados en el sitio más apropiado para la información.
  - Permite el procesamiento paralelo sobre una relación.
- La Fragmentación Vertical y Horizontal pueden ser combinadas
  - Los fragmentos pueden ser sucesivamente fragmentados en cualquier versión.

Transparencia de Datos

- Es el grado en el cual un usuario no tiene noción o no es afectado por los detalles en *como fueron distribuidos* y *donde están almacenados* los datos en un sistema distribuido.
- La Transparencia de Datos puede referirse a:
  - Transparencia de Fragmentación:
    - No se exige que los usuarios conozcan como están fragmentadas las relaciones.
  - Transparencia de Replica
    - Los usuarios no tienen noción de si datos con los que están trabajando están replicados o no.
  - Transparencia de Ubicación
    - No se exige a los usuario que conozcan la ubicación física de las relaciones.
    - *Problema: Nombre único de los items de datos.*

Nombre de los Datos

1. Cada Ítem de datos (relación o fragmento) debe tener un nombre único dentro del sistema
2. Debe ser posible encontrar la localización de un ítem de dato eficientemente.
3. Debe ser posible cambiar la localización de los ítems de datos transparentemente.
4. Cada sitio debe ser capaz crear ítems de datos autónomamente.

- Una solución sería unas un Servidor de Nombre Centralizado
  - Asigna los nombres a todos los ítems de datos
  - Cada sitio mantiene un registro de los ítems que contiene
  - Los sitios preguntan al Servidor de Nombres para localizar los ítems de datos no locales
  - Ventajas:
    - Satisface los puntos 1 a 3
  - Desventaja
    - No Satisface el punto 4
    - El servidor de nombre puede convertirse en un cuello de botella.
    - El servidor de nombre es un solo punto de fallo.

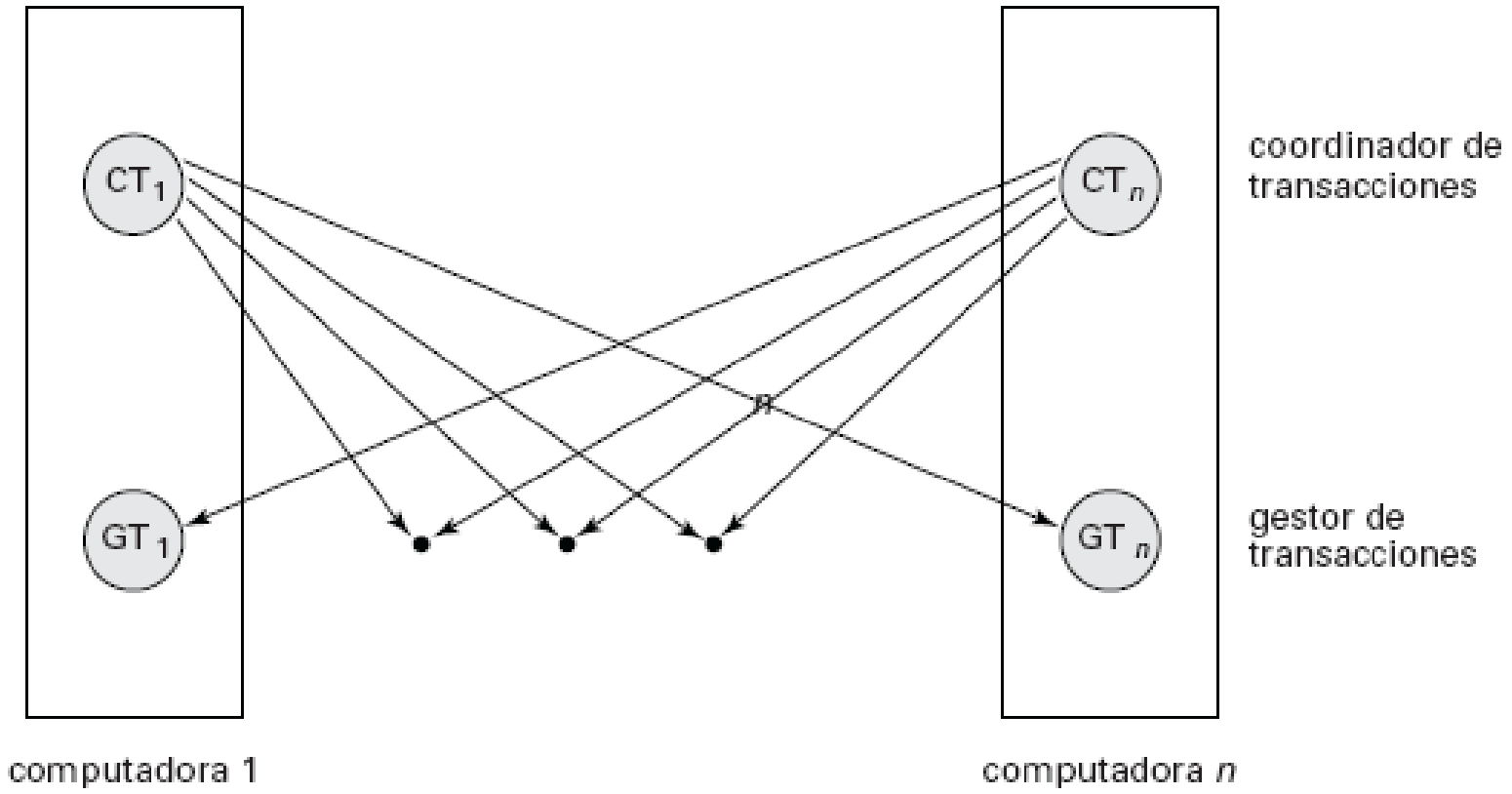
Alias

- Una alternativa al uso de un Servidor de Nombres Centralizado es que cada sitio pueda prefijar su nombre al nombre sus ítems de datos.
  - Permite tener nombre de ítems de datos únicos
  - Se falla en conseguir transparencia de ubicación
- Solución es crear alias para los ítems de datos, cada sitio guarda el mapeamiento de sus alias con los nombre reales.
  - Así el usuario no es afectado ni por la localización física de los ítems de datos ni por el movimiento de los datos de un sitio a otro.

Transacciones Distribuidas

- Las Transacciones pueden acceder a datos ubicados en diferentes sitios
- Cada sitio posee:
  - **Administrador de Transacciones Local:**
    - Mantener una bitácora de la operaciones realizadas para propósitos de recuperación.
    - Coordinar las ejecución concurrente de las transacciones locales y globales de su sitio.
  - **Coordinador de Transacciones:**
    - Iniciar la ejecución de las transacciones distribuidas iniciadas en su sitio
    - Dividir la transacción distribuida en subtransacciones y distribuirlas a los sitios de ejecución correspondiente
    - Coordinar la terminación de la transacciones distribuidas iniciadas en el sitio.

Transacciones Distribuidas



Transacciones Distribuidas

- Los SBDD pueden sufrir los mismos tipos de fallos que los Sistemas de Bases de Datos Centralizados (SBDC) tales como:
  - Fallos de Sw.
  - Fallos de Hw
  - Fallos de Disco
- Sin embargo los SBDD son susceptibles también a otros tipos de fallos
  - Fallo de un sitio
  - Perdida de Mensajes
  - Fallos de los Enlaces de Comunicaciones
  - División de la Red

- Los Fallos de un sitio y la división de la red son por lo general indistinguibles.

Protocolos de Compromiso

- Como toda transacción, para las transacciones distribuidas se deben asegurar las propiedades ACID
- En un SBDD, los protocolos de compromiso tienen como objetivo asegurar la atomicidad de las transacciones.
  - Una Transacción Distribuida debe ser *o comprometida o abortada* en todos los sitios en donde se ejecuto.
  - No es aceptable que una Transacción distribuida confirme en uno a mas sitios y aborte en uno u otros.
- Los protocolos de compromiso más utilizados son
  - Protocolo de Compromiso de dos fases (C2F).
  - Protocolo de Compromiso de tres fases (C3F), protocolo mas costoso pero que evita falencias del C2F.

Protocolo C2F

- Esta basado en el modelos de fallo-parada
  - Los sitios en donde falla una transacción simplemente terminan la ejecución de la misma y resuelven los problemas de consistencia.
- La ejecución del protocolo es iniciada por el coordinador luego de la ejecución de la ultima sentencia de la transacción.
- El protocolo envuelve a todos los sitios en donde tuvo lugar la ejecución de la transacción.
- Sea Si el sitio donde inicio T, entonces Ci es el coordinador de T.
- Sea Sk un sitio en donde se ejecutó parte de T, entonces se dice que Sk es un participante del protocolo de confirmación de T.

C2F – Fase 1

- Esta fase comienza luego de la ultima sentencia de T
- Ci añade el mensaje <Preparar T> a la bitácora y fuerza a guardar la bitácora en almacenamiento persistente.
- Ci envía el mensaje <Preparar T> a todos los participantes Sk's.
- En cada sitio participante Sk:
  - El gestor GTk determina si se puede comprometer su parte de T
  - Si no es posible comprometer T en Sk:
    - GTk añade <No T> a su bitacora
    - GTk responde a Ci envianto el mensaje <Abortar T>
  - Si es posible comprometer T en Sk:
    - GTk añade <Preparado T> a su bitacora y fuerza su grabación.
    - GTk responde a Ci enviando el mensaje <Preparado T>.

C2F – Fase 2

- Esta fase comienza :
  - Cuando Ci recibe todas las respuestas <preparado T> de los Sk's,
  - O cuando ha transcurrido cierto tiempo desde que se ha enviado el mensaje <Preparar T> y no se han recibido todas la respuestas.
- Si se han recibido todas las repuestas de los Sk's y todos son el mensaje <Preparado T>
  - Ci añade el registro <Comprometido T> a la bitácora y fuerza su grabación.
  - Ci envía el mensaje <Comprometer T> a todos los Sk's.
  - Cuando un Sk recibe el mensaje <Comprometer T>
    - Compromete su parte de la transacción T
    - Añade el registro <Comprometido T> a su bitácora y fuerza su grabación
- Si no se han recibido todas las repuestas de los Sk's o una de ellas es <Abortar T>:
  - Ci añade el registro <Abortado T> a la bitácora y fuerza su grabación.
  - Ci envía el mensaje <Abortar T> a todos los Sk's.
  - Cuando un Sk recibe el mensaje <Abortar T>
    - Aborta su parte de la transacción T y ejecuta la recuperación.
    - Añade el registro <Abortado T> a su bitácora y fuerza su grabación

C2F – Fallo de un sitio

- Si un sitio Sk cae antes, durante o después del protocolo C2F, al recuperarse este utiliza su bitácora para determinar el destino de T:
  - Si contiene <Comprometer T>, Sk realiza *rehacer (T)*
  - Si contiene <Abortar T>, Sk realiza *deshacer (T)*
  - Si contiene <Preparado T>, Sk consulta con Ci para determinar el destino de T
    - Si Ci responde <Comprometido T>, Sk realiza *rehacer (T)*
    - Si Ci responde <Abortado T>, Sk realiza *deshacer (T)*
  - Si su bitácora no contiene ningún registro de T
    - La falla de Sk ocurre antes del protocolo de compromiso de T, por lo tanto Ci ha abortado T
    - Sk realiza *deshacer(T)*

C2F – Fallo del Coordinador

- Si el Coordinador falla durante la ejecución del protocolo de confirmación de T, los sitios participantes deben decidir el destino de T:
  - Si un sitio Sk contiene el registro <Comprometer T>, entonces T debe ser comprometida
  - Si un sitio Sk contiene el registro <Abortar T>, entonces T debe ser abortada
  - Si ningún sitio contiene el registro <Preparado T>, entonces Ci no ha decidido que T debe confirmarse. De esta forma T se aborta.
  - Si no ocurren los casos anteriores, entonces todos o algunos de los sitios tienen en su bitácora <Preparado T> y están esperando por la orden del Coordinador. Los sitios deben esperar a que Ci se recupere para conocer el destino de T. **(Problema del Bloqueo).**

C2F – Fallo por división de la Red

- Si el coordinador y todos los sitios participantes quedan en la misma partición, el fallo de la red no tiene efecto en el protocolo
- Si el coordinador y los sitios quedan en varias particiones:
  - En la partición del coordinador se percibe el problema como fallo de uno o más sitios
  - En las particiones que no tienen enlace con el coordinador el problema se percibe como fallo del coordinador
    - No se producen resultados erróneos pues lo máximo que puede ocurrir es que los sitios esperen a contactar nuevamente con el coordinador.
- Nuevamente se percibe que el mayor inconveniente que se puede dar con el protocolo C2F es el **problema del bloqueo**

Recuperación de Transacciones

- Las transacciones dudosas contienen un registro <Preparado T> pero no tienen en ningún sitio un registro <Comprometido T> ni <Abortado T> en su bitácora.
- Los sitios determinan el estado de la transacción contactando con los demás sitios y finalmente esto puede llevar al **problema del bloqueo**
- Un sitio recuperado no debería poder continuar hasta que se hayan confirmado o abortado todas la transacciones recuperadas. Lo cual dejaría indisponible al sitio aun después de superado el fallo.
- Para evitar esto los registros de las transacciones distribuidas pueden llevar una lista de los items de datos que han bloqueado L. <Preparado T, L>
- Al recuperarse el sitio, también se recuperan estas transacciones con sus respectivos bloqueos de datos. Estas transacciones quedan suspendidas hasta que se determinen sus destinos.
- Cualquier nueva transacción que requiera de un dato bloqueado quedara suspendida.

Control de Concurrencia en BDD

- Se utilizan los esquemas de control de concurrencia de sistemas centralizados modificados para BDD
- Se asume que todos los sitios participan del protocolo de compromiso para asegurar así la atomicidad global de las transacciones.
- Se asume inicialmente que todas la replicas son actualizadas.

Concurrencia: Gestor único de bloqueos

- El sistema mantiene un único gestor de bloqueos que reside en un sitio Si
- Todas las peticiones de bloqueo y desbloqueo se realizan en Si
  - Cuando una transacción necesita bloquear un dato, envía una solicitud a Si
    - Si el bloqueo puede concederse se envía un respuesta al sitio.
    - Si el bloqueo no puede concederse, se retrasa la solicitud hasta que se pueda.
  - Una transacción que posea el bloqueo de un dato puede leerlo de cualquier sitio.
  - Las operaciones de escrituras deben realizarse en todos los sitios.
  - Para liberar un bloqueo se envía un mensaje de desbloqueo a Si

Concurrencia: Gestor único de bloqueos

- Este esquema tiene las siguientes ventajas
  - Implementación sencilla.
  - Tratamiento simple de interbloqueos
- Los inconvenientes son:
  - Si puede es un potencial *cuello de botella*
  - El protocolo es vulnerable a fallas de Si
    - Se debe detener el procesamiento de transacciones
    - Se debe tener un gestor de bloqueos de respaldo

Concurrencia: Control de bloqueo distribuido

- Cada sitio posee un administrador de bloqueos
  - Controlan el bloqueo de los datos locales
  - Se necesitan protocolos especiales para tratar con datos replicados.
- Ventaja
  - El sistema es más tolerable a fallos.
- Desventaja
  - La detección de interbloqueos es más complicada

- Varias variantes
  - Copia Primaria
  - Protocolo de Mayoría
  - Protocolo Sesgado
  - Quórum de consenso.

Bloqueos distribuidos: Copia Primaria

- Una replica es seleccionada como Copia Primaria
  - El sitio que contiene la Copia Primaria es llamado Sitio Primario
  - Diferentes datos pueden tener diferentes Sitios Primarios
- Cuando una transacción necesita bloquear un dato Q, esta pide el bloqueo al sitio primario de Q.
  - Si el bloqueo esta disponible se concede el bloqueo, sino se retrasa.
  - Un bloqueo es concedido sobre todas la replicas.
- La solicitud de desbloqueo se envía al sitio primario de Q

- Ventaja:
  - El control de concurrencia para los datos replicados es similar al de datos no replicados – Implementación Sencilla.
- Desventaja:
  - Si el sitio primario de Q falla, Q es inaccesible, aun cuando posea copias accesibles.

Bloqueos distribuidos: Protocolo de Mayoría

- Los administradores de bloqueos de cada sitio administran los bloqueos y desbloqueos de los todos los datos en su sitio.
- Si el datos Q esta replicado en N sitios, la solicitud de bloqueo de Q debe ser enviada a más de la mitad de los N sitios.
- Una transacción no puede operar sobre Q hasta que no haya obtenido un bloqueo en la mayoría de las replicas.

- Ventaja
  - Protocolo valido aun ante falla de los sitios.
- Desventaja
  - Requiere de muchos mensajes, 2(N/2 + 1) para bloqueos y (N/2 + 1) para desbloqueos.
  - Problema de interbloqueo de solicitudes de bloqueo.

Bloqueos distribuidos: Protocolo Sesgado

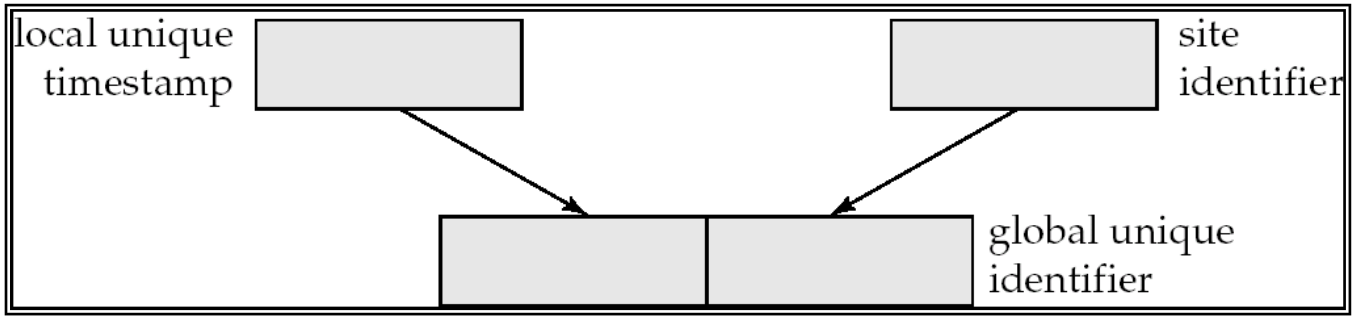
- Cada sitio posee un administrador de bloqueos para los datos en el mismo.
- Las peticiones de bloqueos compartidos y exclusivos se manejan de forma diferente.
  - Bloqueo Compartido
    - Para bloquear un ítem Q en modo compartido solo realiza la solicitud en cualquier sitio donde haya una replica de Q
  - Bloqueo Exclusivo
    - Para bloquear Q en modo exclusivo se hace la solicitud a todos los sitios en donde este replicado Q.
- Ventaja
  - Menos sobrecarga en las operaciones de lectura.
- Desventaja
  - Mayor sobrecarga en las operaciones de escritura.
  - Necesidad de detección de interbloqueos distribuida

Bloqueos distribuidos: Quórum de Consenso

- Es una generalización de los protocolos de Mayoria y Sesgado.
- Cada Sitio tiene asignado un peso
  - Siendo el S la suma de todos los pesos de los sitios.
- Para cada dato Q se escogen dos valores
  - Qr, que es el quórum de lectura
  - Qw, que es el quórum de escritura
  - Se debe cumplir la siguiente condición:
    - Qr + Qw > S y 2*Qw > S
- Si un sitio necesita realizar una lectura debe bloquear tantas replicas tal que la suma de sus pesos sea mayor o igual a Qr
- Si un sitio necesita realizar una escritura debe bloquear tantas replicas tal que la suma de sus pesos sea mayor o igual a Qw
- Se puede asumir que cualquier replica puede ser escrita.

Marcas Temporales

- Aplica la idea de las Marcas Temporales para Transacciones en sistemas distribuidos
- Cada Transacción posee un marca temporal con la cual se determina si cada operación de la transacción es secuencial en el tiempo.
  - Una transacción que llega tarde en el orden secuencial debe ser abortada.
- El problema es como generar marcas temporales unicas en una forma distribuida.
  - Cada sitio genera un marca temporal local usando un contador lógico
  - La Marca de tiempo global es obtenida concatenando el identificador del sitio a la marca local



Marcas Temporales

- Un sitio con lenta generación de marcas pude sufrir inconvenientes
  - La secuencialidad no se ve afectada
  - Las transacciones sería “desaventajdas”, pues podrían llegar siempre tarde a sus operaciones.
- Solución
  - Cada sitio Si define un reloj lógico LCi
  - En todos los sitios Si, cada vez que se vea un transacción T con marca <X, Y> en donde X (valor del reloj para T) sea mayor a LCi, Si hace que su reloj sea igual a X+1… LCi = X+1.
  - Entonces todos los sitios Si tendrán sus relojes lógicos actualizados a la mayor marca de tiempo que han visto.

Replicación de Consistencia Débil

- Muchos SGBD permiten la replicación de relaciones con consistencia débil.
  - Sin garantizar la secuencialidad.
  - Sin garantizar la actualización de los datos.
- Las replicas son una versión consistente de las relaciones
- **Replicación Maestro-Esclavo**
  - La propagación de cambios no es parte de las transacciones
    - Pueden realizarse inmediatamente luego de la confirmación.
    - Pueden ser periódicas
  - En las replicas esclavas solo se realizan lecturas
    - No se requieren de bloqueos.
  - Utilizado en entornos de distribución de información

Replicación de Consistencia Débil

- **Replicación Multimaestro**
  - Las actualizaciones son permitidas en cualquier replica.
  - Las actualizaciones se propagan automáticamente a las demás replicas
  - Modelo Básico usado en BDD soportadas por SGBD comerciales
    - Se realiza luego del protocolo C2F, asegurando la concurrencia a través de una de las técnicas anteriores, por lo general el protocolo sesgado con actualización inmediata.
- También pude darse la **propagación perezosa**
  - Las actualizaciones se dan luego de que la transacción a sido comprometida.
  - Se permite que las actualizaciones ocurran aun ante ausencia de sitios
    - Pero a costa de una menor grado de consistencia

Tratamiento de Interbloqueos

- Considere las dos transacciones siguientes, donde X y T_1 están en el sitio 1 e, Y y T_2 están en el sitio 2

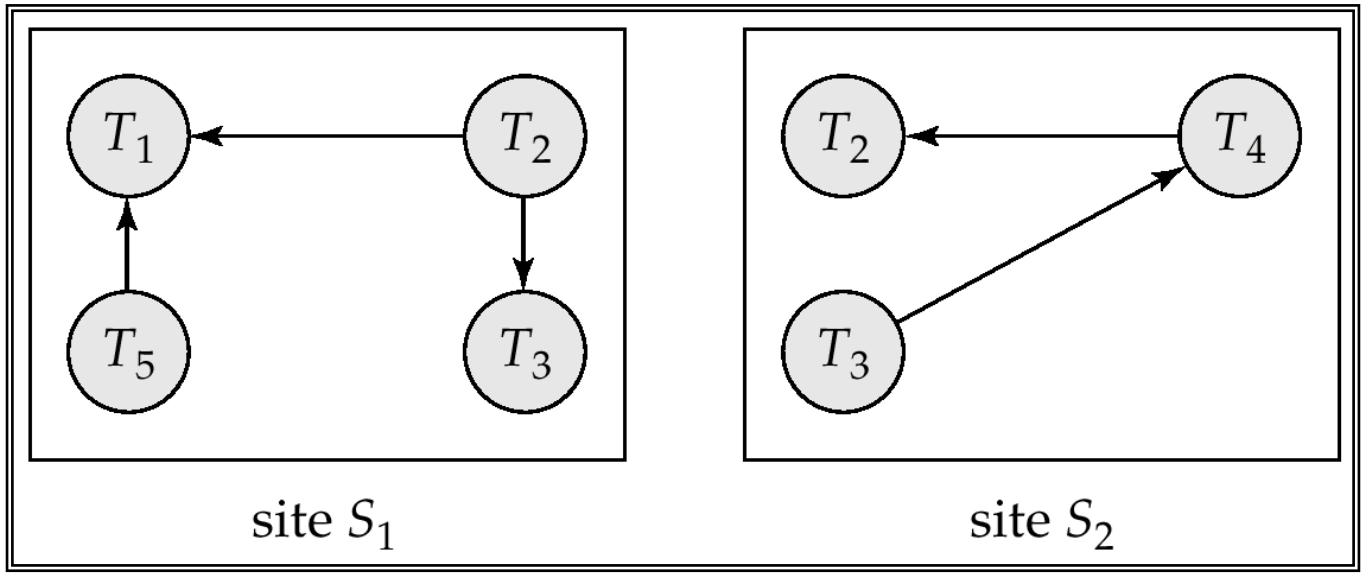
T_1: write (X) write (Y)	T_2: write (Y) write (X)
X-lock on X write (X)	X-lock on Y write (Y) wait for X-lock on X
Wait for X-lock on Y	

EL INTERBLOQUEO NO PUEDE SER DETECTADO LOCALMENTE EN NINGUN SITIO

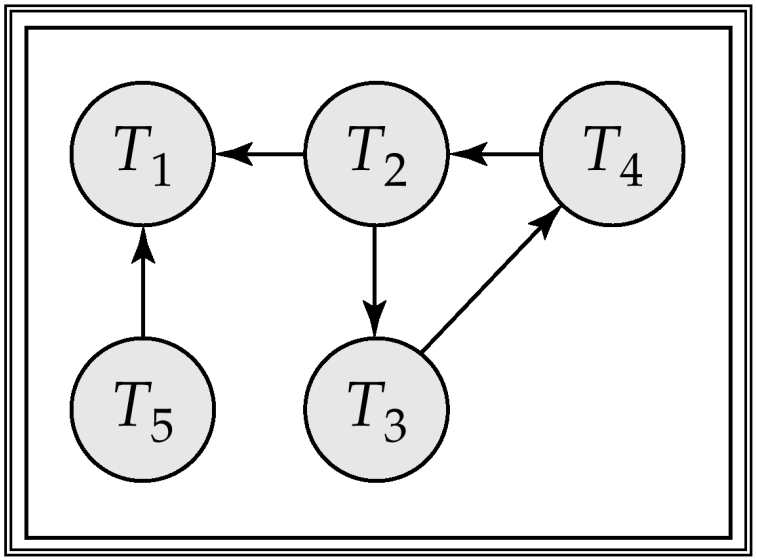
Detección de Interbloqueos: Enfoque Centralizado

- Se construye un grafo de esperas en un sitio: El administrador de interbloqueos.
  - Aproximación al grafo real.
  - Los grafos deben ser *correctos*
    - Si hay algún interbloqueo, se comunique con prontitud
    - Si el Sistema comunica algún interbloqueo, realmente exista un estado de interbloqueo
- El grafo de esperas global puede ser construido o actualizado cuando:
  - Un arco es insertado o removido de algún grafo de esperas local (grafo del sitio).
  - Periódicamente, cuando han ocurrido un cierto número de cambios en un grafo local
  - Siempre que se necesita invocar al algoritmo de detección de ciclos
- Cuando se detecta un ciclo:
  - El coordinador selecciona una víctima para retrocederla e informa a todos los sitios
  - Los sitios retroceden la transacción víctima.
- Este enfoque puede producir retrocesos innecesarios en los casos de:
  - Ciclos Falsos
  - Retroceso no relacionado

Detección de Interbloqueos



Local

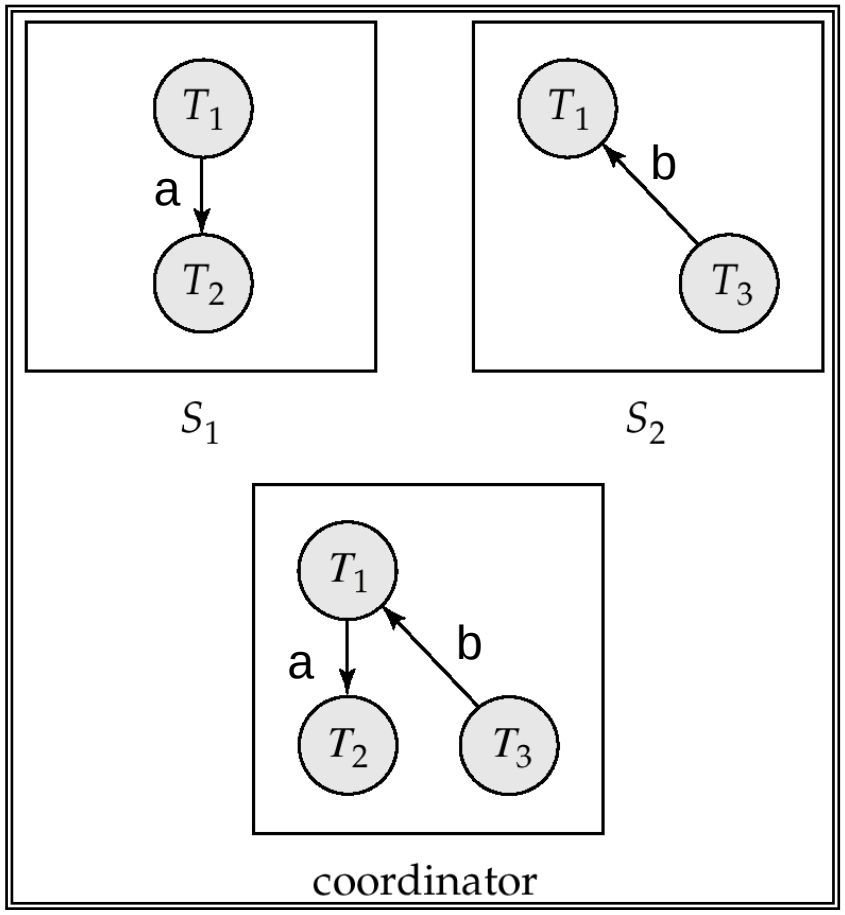


Global

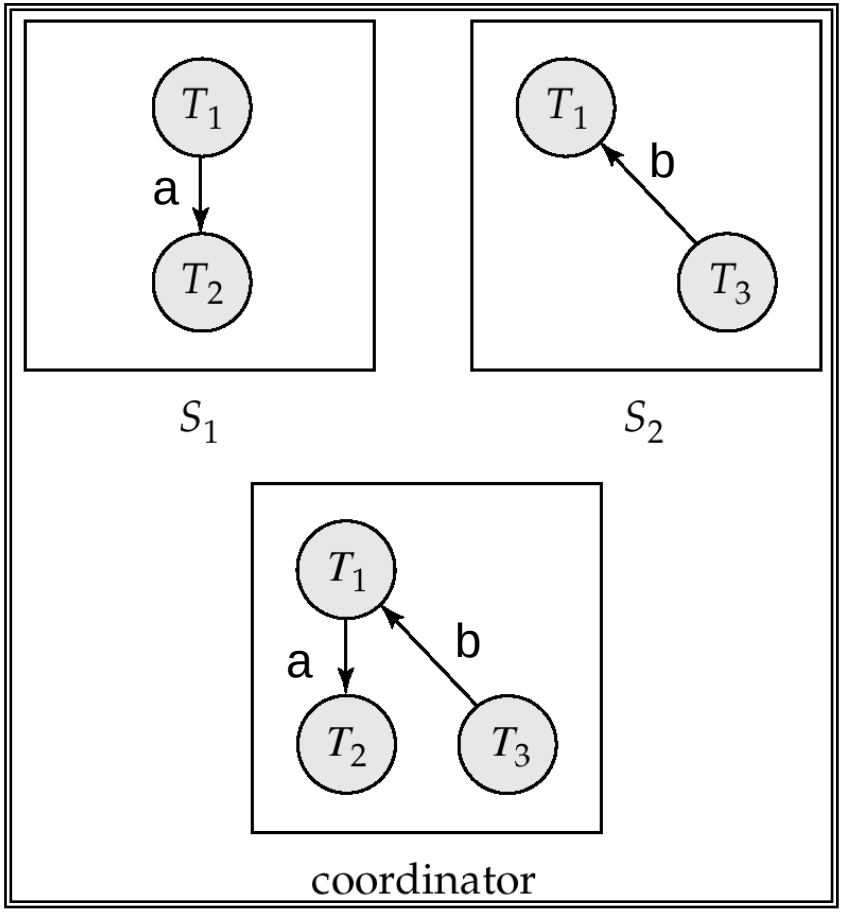
La víctimas pueden ser T2, T3 y T4

Detección de Ciclos: Ciclos Falsos

Estado Inicial



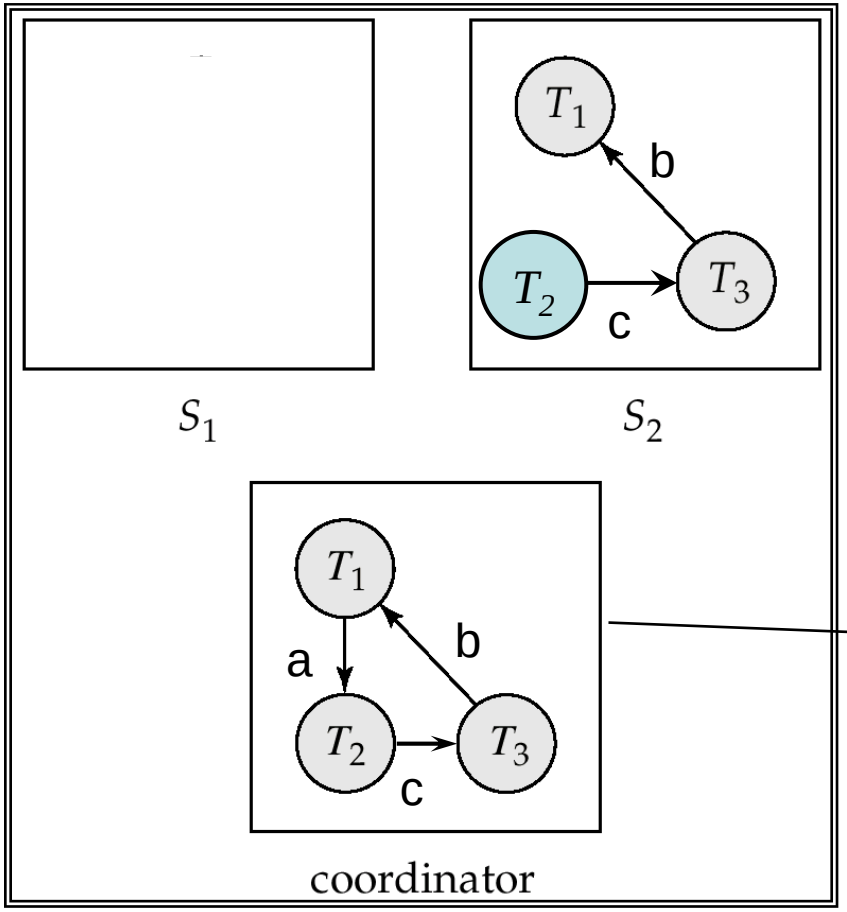
Detección de Ciclos: Ciclos Falsos



Luego ocurre en el sgte. orden que:

1. T2 libera el dato *a*
2. S1 envía el mensaje de eliminar el arco T1 → T2
3. T2 solicita un bloqueo sobre c en el sitio S2, pero T3 tiene bloqueado el dato c
4. S2 envía el mensaje de añadir el arco T2 → T3

Detección de Ciclos: Ciclos Falsos



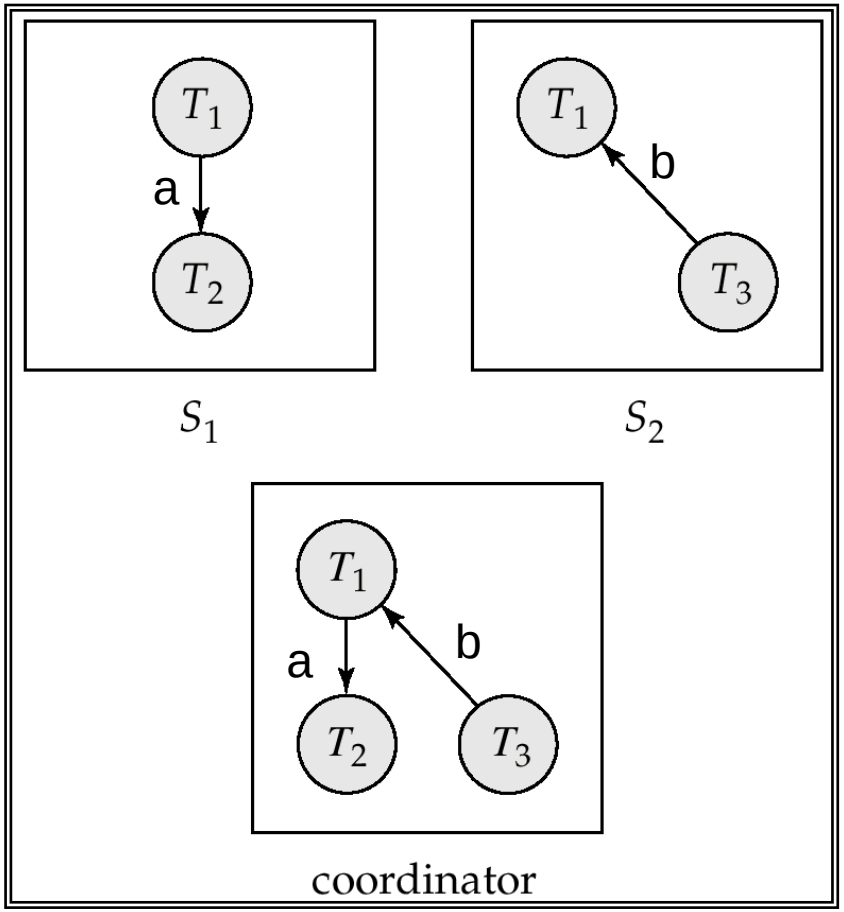
Se detecta un falso ciclo si:

1. El mensaje de S1 llega luego del mensaje de S2
2. El mensaje de S1 se pierde.

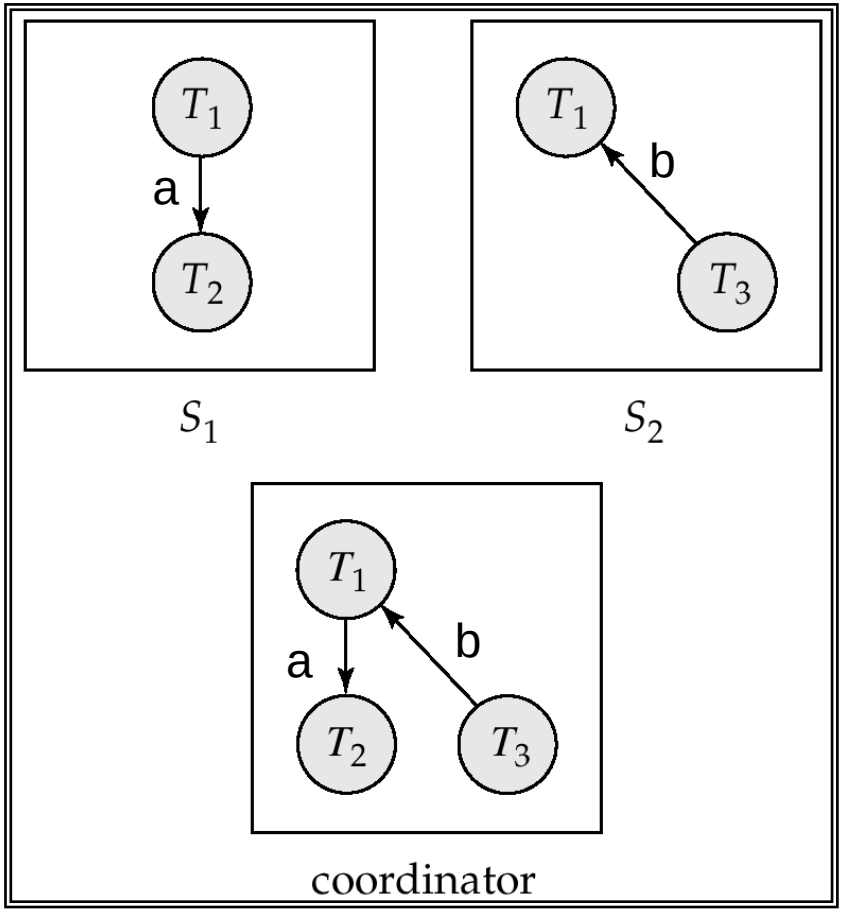
No Real

Detección de Ciclos: Retroceso No Relacionado

Estado Inicial



Detección de Ciclos: Retroceso No Relacionado

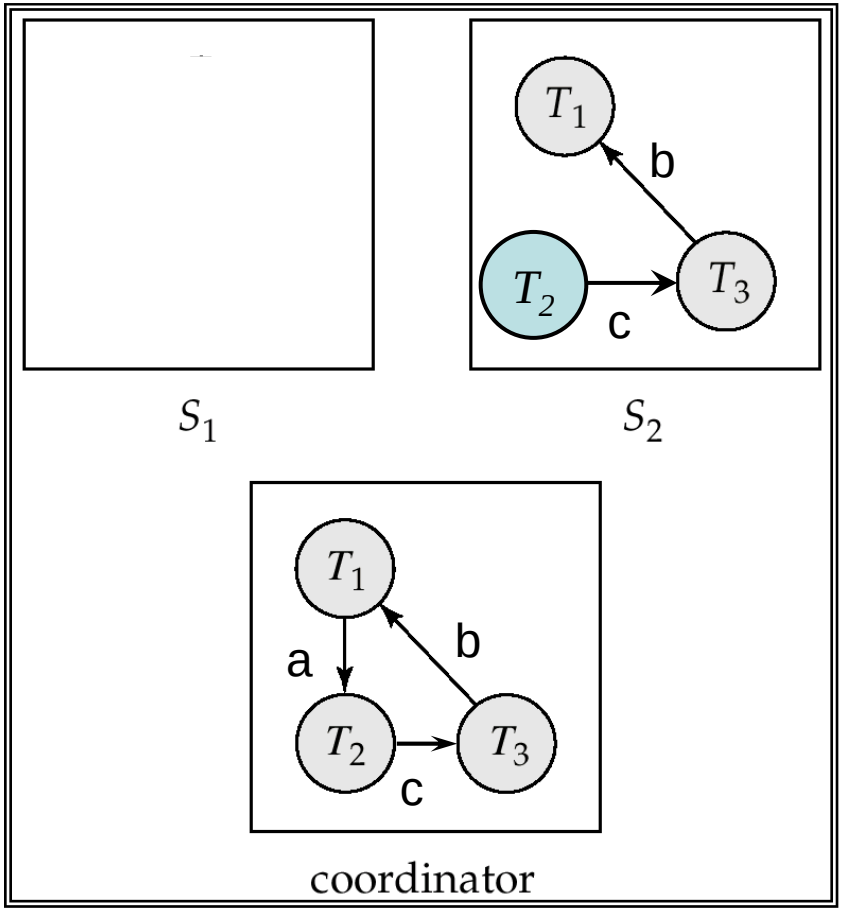


Luego ocurre en el sgte. orden que:

- T2 solicita un bloqueo sobre c en el sitio S2, pero T3 tiene bloqueado el dato c
- S2 envía el mensaje de añadir el arco T2 → T3

o

Detección de Ciclos: Retroceso No Relacionado



Se detecta un falso ciclo si:

1. En S1 se aborta T1 por algún otro problema no relacionado a la detección de ciclos.

Ocurre que:

1. El coordinador pudo haber elegido como víctima a T2 o a T3
2. Se Retrocede innecesariamente a T2 o a T3.

Disponibilidad

- Uno de los objetivos principales de las BDD es conseguir una alta disponibilidad de datos.
- **Alta Disponibilidad:** La mayor parte del tiempo el sistema debería estar disponible (ej: 99,99 %).
- **Robustez**: Capacidad de seguir funcionando aun ante fallos.
- **Problema**: Los fallos son más probables a medida que la BDD crece.
- Para que un SBDD sea robusto debe poder:
  - Detectar los fallos (parte más difícil)
  - Reconfigurar el sistema para continuar con los cómputos
  - Recuperar y/o reintegrar los puntos de fallo reparados.

Reconfiguración

- Re adaptar el sistema para continuar con el computo
- Debe basarse en:
  - Abortar las transacciones que estaban activas en el sitio que ha fallado.
    - Evitando que esta interfieran con nuevas transacciones por mantenerse los bloqueos.
    - Sin embargo, puede ser posible que ciertas transacciones puedan continuar en otros sitios donde existan replicas de los datos.
  - Si el sitio que falla posee replicas de datos, el catalogo del sistema debe ser modificado para que no enviar transacciones a ese sitio hasta que se recupere
  - Si ha fallado un sitio que hospedaba un servidor central de algún subsistema, debe **elegirse** un nuevo sitio que los reemplace.
    - Servidor de Nombre
    - Servidor de Copia Primaria
    - Coordinador de Concurrencia
    - Coordinador de interbloqueos

Reconfiguración

- Dado que una división de la red puede ser indistinguible con el fallo de un solo sitio, debe prevenirse que:
  - Que se elijan dos o mas servidores centrales en una partición.
  - Que en más de una partición se actualice un ítem de dato replicado.
- **Problema**: Las actualizaciones deben continuar aun ante fallos.

Enfoque Basado en Mayoría

- El protocolo de control de concurrencia basado en mayoría puede ser modificado aun para que funcione si hay sitios no disponibles:
  - Cada replica de cada ítem de dato puede tener un **número de versión** que es actualizado cuando el dato es actualizado.
  - Como las peticiones de bloqueo deben enviadas a por lo menos la mitad de los sitios, las operaciones solo continúan si se obtienen permisos de tal cantidad de sitios.
    - Así los datos solo se actualizan en particiones donde exista mayoría de sitios.
  - En las operaciones de lectura, los sitios deben mirar el numero de versión y leer la replica de mayor versión.
    - Pueden haber sitios recién recuperados que tengan numero de versión anterior
    - Adicionalmente, una vez concedidas, en las operaciones de lectura se pueden actualizar los datos en los sitios de menor versión sin necesitar de bloquear los datos.

Enfoque Basado en Mayoría

- Modificación del Protocolo de Mayoría (Cont.):
  - Las operaciones de escritura:
    - Deben buscar la replica de mayor versión y establecer como nueva versión de esta mayor versión incrementada en 1
    - Las escrituras deben realizarse en todas la replicas bloqueadas.
  - Los fallos no causan problemas si:
    - Los sitios en compromiso contengan siempre la mayoría de las replicas.
    - Durantes las lecturas se lean siempre las replicas de mayor versión.
- La reintegración en este caso es trivial.
  - No se realiza ninguna acción particular.

Reintegración de Sitios

- Cuando un sitio se recupera, este debe realizar un procedimiento de actualización de sus datos.
  - Problema: Pueden ocurrir actualizaciones paralelas mientras un sitio se esta actualizando.
  - Solución 1: Detener todas las actualizaciones en el sistema mientras se reintegra un sitio
    - Puede ser una situación inaceptable en el entorno.
  - Solución 2:
    - Bloquear todas la replicas que estén en el sitio
    - Actualizar las replicas
    - Liberar los bloqueos

Selección de Coordinadores

- Coordinadores de Respaldo
  - Sitios que mantienen información suficiente para asumir como coordinador si el coordinador actual falla
  - Debe ejecutar el mismo algoritmo y mantener la misma información interna que el coordinador actual.
  - Permite una rápida recuperación pero a costa de un incremento de procesamiento adicional.
- Algoritmos de elección
  - Tienen como objetivo elegir un nuevo coordinador cuando el actual falla.

Algoritmo Luchador

- Cuando un sitio S_i envía un mensaje al coordinador y este no responde en un intervalo de tiempo T, S_i que el coordinador ha fallado y el trata de convertirse en el nuevo coordinador.
- S_i envía un mensaje de elección a todos los sitios con mayor identificación y espera un tiempo T por una respuesta.
  - Cuando no hay respuesta, S_i asume que todos los sitios de mayor identificación han fallado y se elige a si mismo como coordinador.
  - Cuando hay una respuesta, S_i aguarda un nuevo tiempo T' para recibir la respuesta del nuevo sitio con mayor identificación que ha sido electo.
    - Cuando no se recibe un mensaje del nuevo coordinador en un tiempo T', entonces S_i reinicia el algoritmo
- Después de que se haya recuperado un sitio, comienza de inmediato la ejecución de este mismo algoritmo.
  - Si no hay ningún sitio activo con un número más elevado que el sitio recuperado, este obliga a todos los sitios menores a permitirle transformarse en el nuevo sitio coordinador, aunque ya haya un coordinador activo con un número más bajo.

Procesamiento Distribuido de Consultas

- En el procesamiento de consultas distribuidas se deben considerar los siguientes factores:
  - Diferentes relaciones pueden residir en sitios diferentes.
  - Múltiples copias de una misma relación pueden distribuirse entre los diferentes sitios.
    - Si la relación está replicada, se debe decidir qué replica utilizar.
  - Una relación puede estar fragmentada en subrelaciones, y estas a su vez, distribuidas.
    - Si la relación está fragmentada, hay que calcular varios productos o uniones para reconstruir la relación.
- Por otra parte, para evaluar un consulta que se plantee en un sitio dado, podría ser necesario transferir datos entre varios sitios.
  - La consideración clave aquí es que el tiempo requerido para procesar la consulta está muy ligado al tiempo que se gasta en la transmisión de los datos entre los sitios.

Transformación de Consultas

- Puede ser importante transformar las consultas en función a la definición de los fragmentos
- Considere la fragmentación horizontal de la tabla *account* :

  $account_1 = \sigma_{branch\_name = \text{“Hillside”}} (account)$

  $account_2 = \sigma_{branch\_name = \text{“Valleyview”}} (account)$
- La consulta:
  - Select * from account where branch_name = ‘Hillside’;

  Se traduce en:

  - $\sigma_{branch\_name = \text{“Hillside”}} (account)$

  Y se transforma en:

  - $\sigma_{\text{branch\_name = “Hillside”}} (account_1 \cup account_2)$

  La cual se optimiza en:

  - $\sigma_{branch\_name = \text{“Hillside”}} (account_1) \cup \sigma_{branch\_name = \text{“Hillside”}} (account_2)$

Transformación de Consultas

- Dado que el fragmento $account_1$ solo tiene tuplas pertenecientes a Hillside es posible eliminar la operación de selección sobre $account_2$.

- El Sistema debe ser capaz de detectar que el resultado de la selección sobre $account_2$ es vacío:
  - $\sigma_{branch\_name = \text{“Hillside”}} (\sigma_{branch\_name = \text{“Valleyview”}} (account)) = Ø$

- Asi la estrategia final es devolver como resultado el fragmento $account_1$

  $account_1 = \sigma_{branch\_name = \text{“Hillside”}} (account)$

Procesamiento de Reuniones Sencillas

- Considerando la siguiente expresión del álgebra relacional:

  Cliente <join> deposito <join> sucursal

- Suponiendo que:
  - Ninguna de las tres relaciones está replicada ni fragmentada
  - La relación *cliente* esté almacenada en el sitio S_1
  - La relación *depósito* esté almacenada en el sitio S_2
  - La relación *sucursal* esté almacenada en el sitio S_3
- Sea S_1 el sitio desde donde se emitió la consulta.
  - El sistema tiene que producir el resultado en el sitio S_1.

Procesamiento de Reuniones Sencillas

- Entre las estrategias posibles para el procesamiento de esta consulta están las siguientes:
  1. Enviar copias de las dos relaciones al sitio S_1.
  2. - Enviar una copia de la relación *cliente* al sitio S_2,
     - Calcular $temp_1$ = *cliente* x *deposito* en S_2.
     - Enviar $temp_1$ de S_2 a S_3 y procesar $temp_2$=$temp_1$ x sucursal.
     - Enviar el resultado de $temp_2$ a S_1.
  3. Diseñar estrategias parecidas a la anterior con los papeles S_1, S_2 y S_3 intercambiados.
- Ninguna de estas estrategias puede ser siempre la mejor
  - En la primera estrategia, si se envían las dos relaciones a S_1, y si estas tienen índices, puede ser necesario volver a crearlos en S_1.
  - La segunda estrategia presenta el inconveniente que hay que enviar una relación potencialmente grande (cliente x deposito) de S_2 a S_3, y por tanto puede darse lugar a más transmisiones de red que en la primera estrategia.
- Entre los factores que se deben tener en cuenta están:
  - El volumen de datos enviados.
  - El costo de la transmisión de los bloques de datos entre los sitios.
  - La velocidad relativa en cada sitio.

Estrategias de Semireuniones

- Cuando se tiene que:
  - r_1 es una relación con esquema R_1 en el sitio S_1
  - r_2 es una relación con esquema R_2 en el sitio S_2
  - Y se desea calcular la reunión r_1 <join> r_2
- Si hay muchas tuplas de r_2 que no se reúnan con ninguna tupla de r_1, el envío de r_2 a $S1$ supone el envío de las tuplas que no pueden contribuir al resultado
- Una estrategia posible para evitar esto es la siguiente:
  1. Calcular $temp_1 = \prod_{R1 \cap R2} (r_1)$ en S_1.
  2. Enviar $temp_1$ a S_2.
  3. Calcular $temp_2 = r_2$ <join> $temp_1$ en S_2
  4. Enviar $temp_2$ a S_1.
  5. Calcular r_1 <join> $temp_2$ en S_1.
     El resultado es el mismo que r_1 <join> r_2.

Estrategias de Semireuniones

- Esta estrategia resulta especialmente ventajosa cuando relativamente pocas tuplas de r_2 contribuyen a la reunión.
- Esta situación se produce generalmente cuando r_1 es resultado de una expresión de álgebra relacional que implique a una selección.
  - En esos casos puede que $temp_2$ tenga significativamente menos tuplas que r_2.
    - Es mas ventajoso calcular y enviar $temp_2$ en vez de toda r_2.
  - El envío de $temp_1$ a S_2 supone un coste adicional.
    - Si solo una fracción de tuplas de r_2 contribuye a la reunión, la sobrecarga del envío de $temp_1$ resulta nuevamente ventajosa a tener que enviar toda r_2.

Calcular la semireunion de:

Sitio 1 – R

A1	A2
1	3
1	4
1	6
2	3
2	6
3	7
3	8
3	9

Sitio 2 – S

A2	A3	A4
3	13	16
3	14	16
7	13	17
10	14	16
10	15	17
11	15	16
11	15	16
12	15	16