



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

## ELEMENTOS DE BASES DE DATOS

Segundo Cuatrimestre 2014

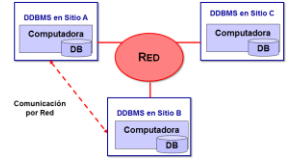
### Clase 22: DDBMS (Parte II)– Gestión de Transacciones Distribuidas

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



## Bases de datos distribuidas - Repaso

- Una **base de datos distribuido** se distribuye de varios **sitios o nodos**.
- Los nodos se conectan a través de la red y pueden variar en tamaño y función.
- Generalmente se ubican geográficamente separados.
- Cada DBMS de un sitio funciona independientemente de los otros.
- Se distinguen:
  - Sistemas de Bases de Datos Distribuidos Homogéneos (SMBDD ó DDBMS).
  - Bases de Datos Distribuidas Heterogéneas



EBD2014\_22 - Mg. Mercedes Vitturini

## SMBDD: Datos distribuidos

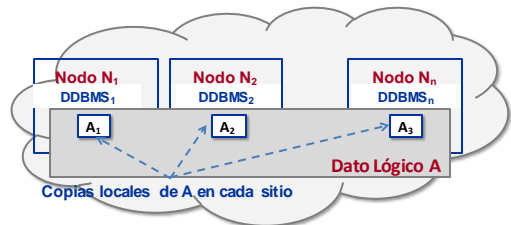
- Replicación** – es posible mantener varias copias (réplicas) idénticas de una relación  $r$  en sitios diferentes
- Fragmentación** – es posible dividir una relación  $r$  en varios fragmentos almacenados en sitios diferentes.
- Replicación y Fragmentación** –

- Se distingue el dato físico o copia del dato lógico como unidad.
- En entornos distribuidos homogéneos con datos replicados la gestión de bloqueos debe realizarse a nivel de dato lógico.

EBD2014\_22 - Mg. Mercedes Vitturini

## Datos Distribuidas

**DATO LÓGICO** y **DATO FÍSICO**



EBD2014\_22 - Mg. Mercedes Vitturini

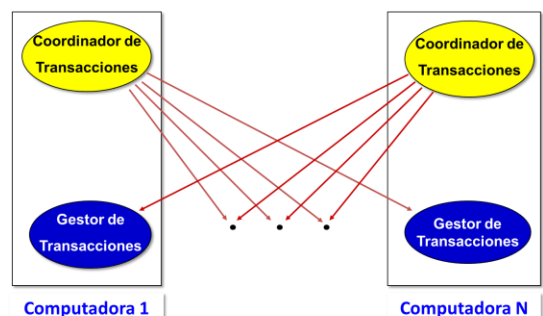
## Transacciones distribuidas

Una transacción eventualmente necesita *acceder a datos de varios sitios*. Para ejecutar transacciones distribuidas cada sitio cuenta:

- **Gestor de transacciones**
    - Administra la bitácora.
    - Coordina la ejecución concurrente de transacciones en el sitio.
  - **Coordinador de transacciones**
    - Inicia la ejecución de transacciones que comienzan en el sitio.
    - Distribuye las subtransacciones entre otros sitios.
    - Coordina la finalización de las transacciones iniciadas, determinando si cometerla o abortarla.
- } Las funciones del DBMS centralizado

EBD2014\_22 - Mg. Mercedes Vitturini

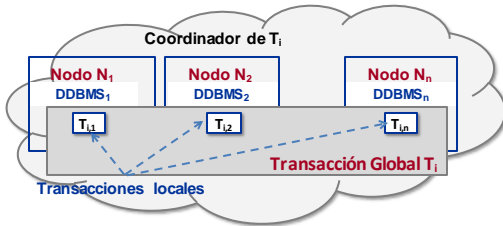
## Gestión de Transacciones Distribuida



EBD2014\_22 - Mg. Mercedes Vitturini

## Transacciones Distribuidas

**TRANSACCIÓN GLOBAL (o lógica)** y **TRANSACCIÓN LOCAL**



EBD2014\_22 - Mg. Mercedes Vitturini

## Transacciones Distribuidas

- En un SMBDD con gestión de transacciones distribuidas se requiere asegurar:
  - **Atomicidad a nivel de transacción global:** una transacción global distribuida *cometerá si y solo si* todas las subtransacciones que la forman están listas para cometer.
  - **Serializabilidad a nivel de transacción global:** a nivel de transacciones globales, la ejecución distribuida debe ser equivalente a una ejecución en serie.

EBD2014\_22 - Mg. Mercedes Vitturini

## Serializabilidad Distribuida

- Una planificación es *serializable* si su efecto es equivalente a una planificación ejecutada en serie a nivel de **transacciones globales**.

**PB2F a nivel de subtransacción:**

- El **protocolo de bloqueo de dos fases** exige que los bloqueos se dividan en dos fases:
  - fase de bloqueo (*crecimiento*) y
  - fase de desbloqueo (*decrecimiento*).
- Se necesita incorporar una modificación a las reglas para que la ejecución sea efectivamente en dos fases.

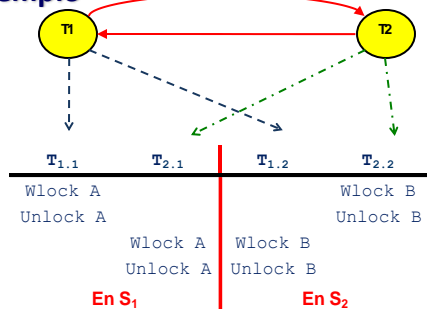
EBD2014\_22 - Mg. Mercedes Vitturini

## Protocolo de Bloqueo de Dos Fases (2PL)

- Supongamos que la transacción global  $T_1$  se divide en dos subtransacciones:
  - ✓  $T_{1.1}$  corre en  $S_1$  y escribe un nuevo valor para A.
  - ✓  $T_{1.2}$  corre en  $S_2$  y escribe un nuevo valor para B.
- Supongamos que la transacción global  $T_2$  se divide en dos subtransacciones:
  - ✓  $T_{2.1}$  corre en  $S_1$  y escribe un nuevo valor para A.
  - ✓  $T_{2.2}$  corre en  $S_2$  y escribe un nuevo valor para B.

EBD2014\_22 - Mg. Mercedes Vitturini

## Ejemplo



**Serializabilidad Local vs. Serializabilidad Global**

EBD2014\_22 - Mg. Mercedes Vitturini

## Serializabilidad Distribuida

- En transacciones distribuidas no es suficiente con **controlar que realicen los bloqueos en dos fases a nivel de subtransacción, sino a nivel de transacción global**.
  - Esto es, **ninguna subtransacción  $T_{n,i}$  de una transacción  $T_n$  debería liberar un bloqueo** si otra subtransacción  $T_{n,j}$  realizará después un bloqueo (*lock-s* ó *lock-x*).
- Ejemplo**
  - En el ejemplo anterior la transacción  $T_1$  viola este principio pues  $T_{1.1}$  libera el bloqueo sobre A antes que  $T_{1.2}$  bloquee a B.

EBD2014\_22 - Mg. Mercedes Vitturini

## Atomicidad Distribuida

- Además para garantizar la **atomicidad global**, es preciso que **TODO**s los sitios en los que se ejecute la transacción  $T$  global coincidan en el resultado final de la ejecución, esto es:
  - Todas las subtransacciones están dispuestas a cometer y la transacción global comente, ó ...
  - al menos una de las subtransacciones no puede cometer y la transacción global aborta.

### ATOMICIDAD Y SERIALIZABILIDAD GLOBAL

- El **coordinador de transacciones** es quien lleva adelante un **protocolo de compromiso** entre los participantes. Vamos a ver:
  - Protocolo de Compromiso Distribuido.
  - Protocolo de Compromiso de Dos Fases.

EBD2014\_22 - Mg. Mercedes Vitturini

## Atomicidad y Serializabilidad Distribuida

### Protocolos de coordinación

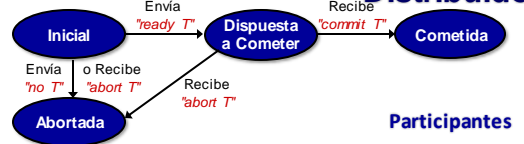


## Protocolo #1: Compromiso Distribuido (PCD)

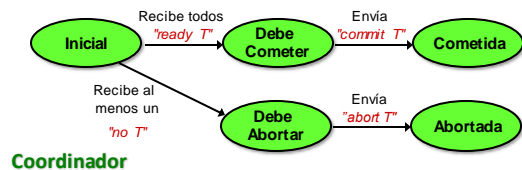
- Sea  $T$  una transacción global iniciada en un sitio  $S$  y dividida en subtransacciones ejecutándose en diferentes sitios (incluido el mismo sitio  $S$ ).
  - El sitio  $S$  oficia de **coordinador**.
  - Las subtransacciones en los distintos sitios son los **participantes**.
- Cada subtransacción  $T_i$  de  $T$  decide por cometer o abortar.
  - Una vez que  $T_i$  decide, envía al coordinador un mensaje "ready  $T$ " o "no  $T$ ".
  - El coordinador toma la decisión final en función de las votaciones de todos los participantes.

EBD2014\_22 - Mg. Mercedes Vitturini

## Protocolo de Compromiso Distribuido



Participantes



Coordinador

EBD2014\_22 - Mg. Mercedes Vitturini

## Características PCD

- Cada subtransacción  $T_i$  decide entre **cometer** o **abortar**.
- Cuando  $T_i$  toma una decisión, envía un mensaje "ready  $T$ " o "no  $T$ ", que representa su voto por cometer o abortar respectivamente.
  - Si  $T_i$  envía un "no  $T$ ", entonces sabe que  $T$  va a abortar.
  - En cambio, si envía un "ready  $T$ " no puede asegurar que  $T$  sea cometida, ya que eso depende de lo que voten los otros participantes.
- Después de votar "ready  $T$ ",  $T_i$  debe esperar la decisión del coordinador.

Participantes y coordinador dejan registro en la bitácora de los mensajes antes de enviarlos

EBD2014\_22 - Mg. Mercedes Vitturini

## PCD – Coordinador

- El coordinador es quien realiza el proceso de decisión:
  - Si **recibe** todos "ready  $T$ " de los participantes  $\Rightarrow$  decide por cometer y **envía** un "commit  $T$ " a todos los participantes.
  - Si **recibe** al menos un "no  $T$ " de alguno de los participantes  $\Rightarrow$  **decide** abortar y envía un "abort  $T$ " a todos los participantes.

EBD2014\_22 - Mg. Mercedes Vitturini

### PCD – Participantes

- Cuando los participantes reciben la orden de cometer, realizan “**el compromiso**” asumido:
    - Escriben y vuelcan los registros de bitácora al almacenamiento no volátil.
    - Liberan los bloqueos.
- } Comete  $T_i$
- En este esquema existe: **mensajes de voto** y **mensajes de decisión**.
  - Como el coordinador también es participante, los mensajes que se envía a sí mismo no requieren tráfico de red.

EBD2014\_22 - Mg. Mercedes Vitturini

### Nuevos Tipos de fallos

- Nuevos fallos que pueden ocurrir en un entornos distribuidos:
  - **Fallo de un sitio**, mientras los demás sitios siguen funcionando.
  - **Pérdida de mensajes**, a cargo del protocolo de comunicación.
  - **Fallo físico de un enlace**, a cargo del protocolo de comunicación.
  - **Partición de la red**, uno o más nodos queda/n ‘aislado/s’ del resto del sistema.

EBD2014\_22 - Mg. Mercedes Vitturini

### Problemas del PCD

- El PCD podría generar fácilmente **estados de bloqueo** ante fallos en los participantes, en la red ó pérdidas de mensajes.
- Así, una subtransacción que votó por  $\langle ready T \rangle$  y no recibe respuesta no puede decidir por si sola cometer o abortar.
- **Ejemplo:**
  - La subtransacción  $T_i$  envió  $\langle ready T \rangle$ . Mientras, mantiene un bloqueo sobre una copia de un ítem de dato A y alcanza su punto de compromiso *dispuesta a cometer*.
  - Espera, y no recibe del coordinador ningún mensaje “*commit T*” o “*abort T*”.  $T_i$  debe permanecer en ese estado, reteniendo el bloqueo sobre la copia local de A.
  - $T_i$  está **bloqueada** (no deadlock!) y no puede decidir por si sola.

EBD2014\_22 - Mg. Mercedes Vitturini

### Protocolo #2: Compromiso de 2 Fases (PC2F)

Sea  $T$  una transacción distribuida que se inicia en el sitio  $S_i$ , y sea  $C_i$  el coordinador de transacciones de esa localidad. Luego de la ejecución parcial  $C_i$  inicia el protocolo de compromiso de 2 fases:

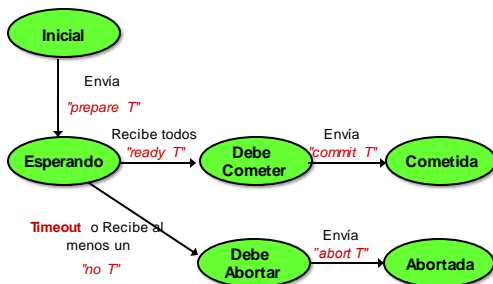
#### Fase 1 - Votación

- $C_i$  agrega a el registro  $\langle prepare T \rangle$  a la bitácora en memoria estable y envía el mensaje “prepare  $T$ ” a los participantes junto con la lista de sitios participantes.
- El gestor de transacciones de cada sitio participante decide su respuesta: si no está en condiciones de resolver su parte, agrega  $\langle no T \rangle$  en su bitácora en memoria estable y envía “no  $T$ ”. Sino agrega  $\langle ready T \rangle$  en su bitácora y envía “ready  $T$ ”.

#### Fase 2 - Decisión

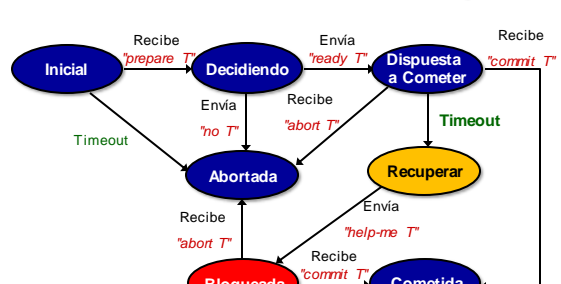
- $C_i$  en función de las respuestas, determinará si la transacción global puede cometer o no.
- Según el veredicto,  $C_i$  agrega en su bitácora  $\langle commit T \rangle$  ó  $\langle abort T \rangle$  y envía el mensaje que corresponda a cada participante.
- Cada participante que reciba el mensaje de  $C_i$  lo graba en bitácora.

### PC2F – Diagrama de Estado Coordinador



EBD2014\_22 - Mg. Mercedes Vitturini

### PC2F – Diagrama de Estado Participante



EBD2014\_22 - Mg. Mercedes Vitturini

## PC2F – Fase 1 (votación)

1.  $C_i$  agrega el registro  $\langle \text{prepare } T \rangle$  a la bitácora y lo graba en memoria estable.
2.  $C_i$  envía un mensaje  $\text{"prepare } T\text{"}$  a todos los sitios donde se ejecutó  $T$ .
3. Cada gestor de transacciones que recibe el mensaje determina si puede o no cometer su porción de  $T$ .
  - **Respuesta Si:** agrega  $\langle \text{ready } T \rangle$  a la bitácora, la graba en memoria estable y envía el mensaje  $\text{"ready } T\text{"}$  a  $C_i$ .
  - **Respuesta No:** agrega  $\langle \text{no } T \rangle$  a la bitácora, la graba en memoria estable y envía el mensaje  $\text{"no } T\text{"}$  a  $C_i$ .

EBD2014\_22 - Mg. Mercedes Vitturini

## PC2F– Fase 2 (decisión)

- $C_i$  recibe respuesta de todos los sitios o vence su *timeout*.
- Si recibe  $\text{"ready } T\text{"}$  de cada sitio que ejecutó una sub-transacción de  $T$ , agrega el registro  $\langle \text{commit } T \rangle$  a la bitácora, lo graba en memoria estable y envía el mensaje  $\text{"commit } T\text{"}$  a todos los sitios participantes.
- En otro caso, se graba el registro  $\langle \text{abort } T \rangle$  en la bitácora, graba en memoria estable y envía el mensaje  $\text{"abort } T\text{"}$  a todos los sitios participantes.

EBD2014\_22 - Mg. Mercedes Vitturini

## PC2F

- Un sitio en el cual  $T$  se ejecutó puede abortar  $T$  en cualquier momento antes de enviar  $\text{"ready } T\text{"}$ .
- El mensaje  $\text{"ready } T\text{"}$  es una promesa a que se cometerá o abortará cuando lo indique el coordinador  $C_i$ . Para cumplir la promesa aún ante la presencia de fallos se almacena información en memoria estable.
- Puesto que se requiere *unanimidad* para cometer una transacción, la suerte de  $T$  se sella tan pronto como un sitio responde  $\text{"no } T\text{"}$ .
- El veredicto final sobre  $T$  lo determina el coordinador mediante un  $\langle \text{commit } T \rangle$  o  $\langle \text{abort } T \rangle$  en la bitácora, que luego además envía por mensaje.

EBD2014\_22 - Mg. Mercedes Vitturini

## PC2F - Fallo de un participante

El coordinador  $C_i$  detecta que un sitio ha fallado:

- Si el sitio participante falla antes de responder  $\text{"ready } T\text{"}$ , el coordinador asume que fue un  $\text{"no } T\text{"}$ .
- Si el sitio participante falla después de responder  $\text{"ready } T\text{"}$ , el coordinador recibe la respuesta, se ignora la falla.

EBD2014\_22 - Mg. Mercedes Vitturini

## Recuperación del Participante

- Cuando el sitio participante se recupera de un fallo consulta su bitácora:
  - Si contiene un  $\langle \text{commit } T \rangle$  ejecuta un  $\text{Redo}(T)$ .
  - Si contiene un  $\langle \text{abort } T \rangle$  ejecuta un  $\text{Undo}(T)$ .
  - Si contiene un  $\langle \text{ready } T \rangle$  debe consultar al coordinador  $C_i$  (enviando un mensaje  $\text{"help-me } T\text{"}$ ) para decidir sobre  $T$ .
    - Si  $C_i$  contesta  $\text{"commit } T\text{"}$ , ejecuta  $\text{Redo}(T)$ .
    - Si  $C_i$  contesta  $\text{"abort } T\text{"}$ , ejecuta  $\text{Undo}(T)$ .
    - Si  $C_i$  no contesta se comunica con los otros sitios.
  - En cualquier otro caso, se ejecuta  $\text{Undo}(T)$ .

EBD2014\_22 - Mg. Mercedes Vitturini

## 2PC – Fallo del coordinador

Si ocurre que el coordinador falla en medio de la ejecución o ciertos participantes quedan incomunicados con el coordinador se pueden dar dos situaciones:

1. Que los participantes se comuniquen entre si y puedan decidir que hacer.
2. Que los participantes deban esperar que se recupere el coordinador.

EBD2014\_22 - Mg. Mercedes Vitturini

### Fallo del Coordinador

Inspeccionar la bitácora de cada participante:

- Si un sitio activo contiene un `<commit T>` en su bitácora, entonces T debe ser cometida.
- Si un sitio activo contiene un `<abort T>` en su bitácora, entonces T debe ser abortada.
- Si algún sitio no contiene un `<ready T>` entonces el coordinador fallado C<sub>i</sub> no pudo haber decidido cometer T ya que no recibió un `"ready T"` de todos los participantes. En general, en este caso se decide abortar T.
- **En otro caso, cada participante debe tener un `<ready T>` en su bitácora y los participantes deben esperar que se recupere C<sub>i</sub>.**

EBD2014\_22 - Mg. Mercedes Vitturini

### Ejercicios

- Supongamos una transacción global T que se inicia en N<sub>2</sub>: T<sub>2</sub> (coordinador) y T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub> y T<sub>5</sub> sincronizados bajo el protocolo de compromiso de dos fases. Explicar el proceso de recuperación en cada caso:
  - T<sub>3</sub>, falla después de enviar el mensaje "no T" al coordinador T<sub>1</sub>, T<sub>2</sub>, T<sub>4</sub> y T<sub>5</sub> votaron "ready T".
  - T<sub>1</sub>, falla antes de decidir; todos los otros participantes votaron "ready-T".
  - T<sub>3</sub>, falla después de enviar el mensaje "ready T" al coordinador T<sub>1</sub>, T<sub>2</sub>, T<sub>4</sub> y T<sub>5</sub> votaron "ready T".
  - Todos votaron "ready-T" y el coordinador T<sub>2</sub> falla después de enviar "commit T" a T<sub>1</sub> y antes de enviar el mismo mensaje al resto de los participante.

EBD2014\_22 - Mg. Mercedes Vitturini

### Equivalencias de Mensajes

Semejanzas de mensajes en Protocolos de Compromiso:

- *Principles of Database and Knowledge-Base Systems.* Jeffrey D. Ullman.
- *Database System Concept.* Abraham Silberschatz, Henry F. Korth, S. Sudarshan.

| Ullman         | Silberschatz  |
|----------------|---------------|
| begin-vote     | "prepare T"   |
| vote-commit    | "ready T"     |
| vote-abort     | "no T"        |
| abort          | "abort T"     |
| commit         | "commit T"    |
| help-me        |               |
| prepare-commit | "precommit T" |

EBD2014\_22 - Mg. Mercedes Vitturini

### Serializabilidad con Estampillas de Tiempo

- También es posible adaptar los protocolos de control de concurrencia basados en estampillas para entornos distribuidos asignando una estampilla global a las subtransacciones.
- En un sistema distribuido, cada transacción global debe tener una *estampilla de tiempo única* para usar en la decisión del orden de serializabilidad.
- Generación de estampillas únicas:
  - Modelo Centralizado.
  - Modelo Distribuido.

EBD2014\_22 - Mg. Mercedes Vitturini

### Generación de Estampillas de Tiempo

**Modelo Centralizado** – un único sitio es encargado de generar y distribuir las estampillas de tiempo. Desventajas:

- Degrada la performance general del sistema.
- Compromete la disponibilidad.

**Modelo Distribuido** – cada transacción tiene una estampilla de tiempo que surge de concatenar la estampilla de tiempo local con el identificador de sitio.

EBD2014\_22 - Mg. Mercedes Vitturini

### Modelo Distribuido para Estampillas de Tiempo

- Importa el orden donde se ubica la estampilla de tiempo local y el identificador de sitio.
  - El identificador de sitio debe ubicarse en el lugar menos significativo para asegurar que no existe un sitio que genera siempre estampillas mayores.
  - El tiempo local se puede representar por un *contador o reloj lógico*.



EBD2014\_22 - Mg. Mercedes Vitturini

## Estampillas de Tiempo: Modelo Distribuido

### Problema de este esquema:

- La dificultad con el modelo distribuido es que un sitio podría generar estampillas de tiempo a mayor velocidad que otros sitios.
  - Esto se evita si cada sitio utiliza un *reloj lógico*, implementado mediante un contador.
  - Para sincronizar los relojes lógicos, se requiere que *cada sitio  $S_i$  avance su reloj lógico cuando arriba una nueva subtransacción a ese sitio que viene de otro sitio con un contador superior.*

EBD2014\_22 - Mg. Mercedes Vitturini

## Replicación con menor rigor en consistencia

- Algunas BD comerciales soportan replicación de datos con grados de consistencia menor.
- **Replicación master-slave:** las actualizaciones son realizadas en un sitio "master" y propagadas a los sitios "slave".

### Características

- La **propagación no forma parte de la transacción:** puede hacerse después del commit o periódicamente.
- Los datos en los sitios slave solo son accedidos en modo lectura y no requieren locks
- Se usan en sistemas con información distribuida y con requerimiento de consultas sólo de lectura.

EBD2014\_22 - Mg. Mercedes Vitturini

## Deadlocks Distribuidos

**RECORDAMOS:** "el sistema" está en *deadlock* si existe un conjunto de dos o más transacciones, tal que " *$T_i$  espera por recurso (dato) que tiene  $T_{i+1}$ ,  $T_{i+1}$  espera por un recurso que tiene  $T_{i+2}$ , ...,  $T_{k-1}$  espera por un recurso que tiene  $T_k$  y  $T_k$  espera por un recurso que tiene  $T_i$ .*"

- Bajo esta situación, ninguna transacción del grupo puede progresar.
- Existen dos formas de tratar deadlocks:
  - **Prevención.**
  - **Detección y Recuperación.**

EBD2014\_22 - Mg. Mercedes Vitturini

## Deadlock Distribuido – Prevención

- Una **técnica de prevención simple** usando la estrategia de bloqueo distribuido (por ejemplo *write-locks-all*) es:
  - Asumir un ordenamiento lexicográfico de los ítems de datos.
  - Si  $A < B$  según el orden anterior, entonces una transacción T debe bloquear todas las copias de A antes de bloquear cualquier copia de B.
  - También las copias de cada ítem A se ordenan y una transacción bloquea todas las copias de A que necesita en el orden predefinido.

EBD2014\_22 - Mg. Mercedes Vitturini

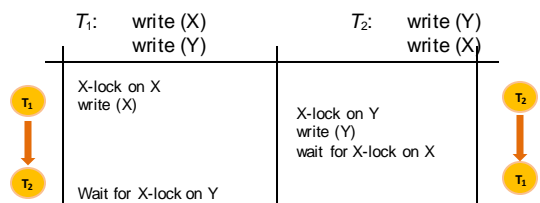
## Deadlock Distribuido - Detección

- Una estrategia para detectar deadlocks es usar **grafos de espera**. En el grafo los nodos representan a las transacciones y/o subtransacciones y los arcos las esperas por datos.
- Cada sitio mantiene su **grafo de espera local**.
- Cuando una transacción  $T_i$  en un sitio  $S_1$  necesita un recurso del sitio  $S_2$ ,  $T_i$  envía un mensaje a  $S_2$ .
- Si el recurso lo tiene una transacción  $T_j$ , **se agrega el arco  $T_i \rightarrow T_j$  al grafo de espera del sitio  $S_2$ .**
- Si el grafo de espera local contiene un ciclo entonces se produjo una situación de *deadlock*.

EBD2014\_22 - Mg. Mercedes Vitturini

## Deadlock Distribuido

Dadas las siguientes transacciones, con el **dato X y la transacción  $T_1$  en  $S_1$** , y **el dato Y y la transacción  $T_2$  en  $S_2$** :



**Resultado: deadlock que no se detecta localmente**

EBD2014\_22 - Mg. Mercedes Vitturini

### Grafo de Espera – Modelo Centralizado

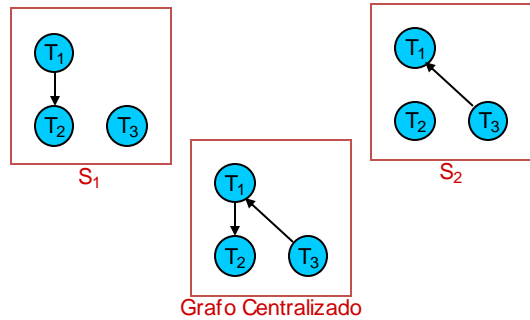
- En el *modelo centralizado*, un mismo sitio se encarga de construir un grafo de espera global.
- Tal sitio se denomina *coordinador de detección de deadlocks*.

**Problema:** existen dos tipos de grafos de espera:

- **Grafo Real** – que describe el estado real aunque desconocido del sistema.
- **Grafo Construido** – es la aproximación generado por el coordinador.
- Las diferencias se producen por demoras de comunicación en el sistema o mensajes perdidos.

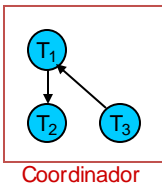
EBD2014\_22 - Mg. Mercedes Vitturini

### Grafo de Espera Global



EBD2014\_22 - Mg. Mercedes Vitturini

### Problemas del Grafo Global



- Supongamos que  $T_2$  libera el recurso que estaba ocupando en  $S_1$  (se elimina la arista  $T_1 \rightarrow T_2$ ) y solicita a  $T_3$  el recurso que está ocupando en  $S_2$  (se inserta la arista  $T_2 \rightarrow T_3$ ).
- Si el mensaje de insertar llega antes que el de eliminar se produce un *ciclo falso* que fuerza iniciar un proceso de recuperación de deadlocks.

EBD2014\_22 - Mg. Mercedes Vitturini

### Grafo de Espera Global

- ¿Cuándo *actualizar* el grafo de espera global?
  - Cada vez que se inserta o elimina un arco en alguno de los grafos de espera locales.
  - Periódicamente, una vez que se haya efectuado cierto número de cambios en un grafo de espera local.
  - Siempre que el coordinador tenga que invocar al algoritmo de detección de ciclos.
- Cuando se invoca al algoritmo de detección de ciclos, el coordinador examina su grafo global.
- Si encuentra un ciclo, se elige una víctima y se realiza un retroceso de la misma, informando de esto a todas las localidades.

EBD2014\_22 - Mg. Mercedes Vitturini

### Grafo de Espera – Modelo Distribuido

- En el *modelo de distribución total*, los sitios comparten equitativamente la responsabilidad de detectar deadlocks.
- Cada sitio contiene un grafo de espera local que representa *una parte del grafo de espera global*.
  - El objetivo es que, si existe un deadlock, aparezca un ciclo en al menos uno de los grafos parciales.
- Cada grafo de espera local agrega un *nodo adicional  $T_{ex}$* , donde:
  - $T_i \rightarrow T_{ex}$  indica que  $T_i$  espera un dato usado por una transacción en otra localidad.
  - $T_{ex} \rightarrow T_j$  indica que una transacción en otra localidad espera por un recurso que tiene asignado  $T_j$ .

EBD2014\_22 - Mg. Mercedes Vitturini

### Modelo de Distribución Total

- Existen dos casos posibles:
  - Si un grafo de espera local contiene un ciclo que no involucra a  $T_{ex}$ , entonces el sistema se encuentra en *deadlock*.
  - Si un grafo de espera local contiene un ciclo que involucra a  $T_{ex}$ , existe la *posibilidad* de que se haya producido un deadlock, no es seguro.
- En el último caso, se invoca a un *algoritmo distribuido de detección de deadlock*.

EBD2014\_22 - Mg. Mercedes Vitturini

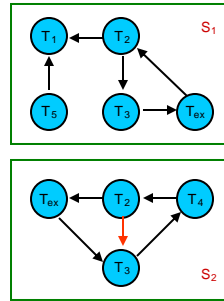


## Detección de Deadlocks Distribuido

- Supongamos que en  $S_1$  existe un ciclo que involucra a una transacción externa  $T_{ex}$  en  $S_2$ .
- $S_1$  envía un mensaje de detección de deadlocks a  $S_2$ , con la información de su grafo local referente al ciclo.
- Cuando  $S_2$  recibe el mensaje actualiza su grafo de espera con la información que recibe.
- Si el grafo de espera actualizado contiene un ciclo que no involucra a  $T_{ex}$ , entonces existe un *deadlock* y se invoca al algoritmo de recuperación.
- Si el grafo de espera actualizado contiene un ciclo que involucra a una transacción externa  $T_{ex}$  en  $S_2$ , se respeta el procedimiento, que en algún momento se detiene ya que la red es finita.

EBD2014\_22 - Mg. Mercedes Vitturini

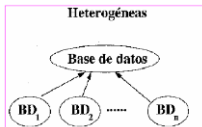
## Detección de Deadlocks



- $S_1$  detecta el posible deadlock  $T_{ex} \rightarrow T_2 \rightarrow T_3 \rightarrow T_{ex}$  de  $S_2$ .
- Notemos que  $T_4$  es externa para  $S_1$ , mientras que  $T_1$  y  $T_5$  son externas para  $S_2$ .
- Como  $T_3$  espera un dato de la localidad  $S_2$ , transmitirá un mensaje a  $S_2$ .
- Cuando  $S_2$  recibe el mensaje actualiza el grafo de espera y obtiene el siguiente ciclo:  $T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_2$ .
- Luego, se produce un deadlock "local" y se invoca al algoritmo de recuperación de deadlocks.

EBD2014\_22

## Bases de Datos Distribuidas Heterogéneas



La heterogeneidad se debe a que los datos de cada BD son de diferentes tipos o formatos. El enfoque heterogéneo es más complejo que el enfoque homogéneo.

## Bases de Datos Distribuidas Heterogéneas

- **BDD Heterogéneas:** Ciertas aplicaciones usan datos de diversas bases de datos preexistentes localizadas en diversas plataformas de HW y SW heterogéneas.
  - Existen diferencias en los modelos de datos conceptuales y lógicos.
  - El manejo de transacciones es diferente e independiente.
- Ejemplos
  - Sistemas inmobiliarios.
  - Análisis de información regional.

EBD2014\_22 - Mg. Mercedes Vitturini

## BDD Heterogéneas

En los últimos años ha crecido el interés por las BDD heterogéneas con las posibilidades que brinda Internet.

### Ventajas

- Preservación de identidad original
  - Hardware
  - Software de base
  - Aplicaciones
- Autonomía y administración local.
- Permite la elección del software de SMBD.

EBD2014\_22 - Mg. Mercedes Vitturini

## BDD Heterogéneas

- Usar un **sistema multi-bases de datos:** es una capa de software encima de los sistemas de bases diseñada para manejar información de bases de datos heterogéneas
  - Crea la ilusión de integración en una única bases de datos lógica sin requerir de la integración física.
- La integración total de bases de datos heterogéneas no siempre es viable. Pueden existir
  - **Dificultades técnicas:** demasiado costoso.
  - **Dificultades organizacionales:** dificultades políticas.

EBD2014\_22 - Mg. Mercedes Vitturini

## BDD Heterogéneas

- Cuestiones a considerar en BDD heterogéneas
  - **Vista unificada de los datos:** definir mecanismos para integrar diversos modelos de datos locales diferentes en un único modelo de datos global.
  - **Procesamiento de consultas:** cómo atender a los requerimientos de consultas integradas en diversas consultas locales.

EBD2014\_22 - Mg. Mercedes Vitturini

## Vista unificada de los datos

- Integrar esquemas diversos en un esquema común: **esquema global.**
- La integración de esquemas no es directa:
  - **Uso de nombres distintos para relaciones/atributos.**  
Ejemplo: los atributos domicilio y dirección.
  - **Nombres iguales de atributos/relaciones con significado diferente.**
    - **Ejemplo:** el atributo contacto en un sistema puede representar un teléfono y en otro el nombre de una persona.
  - **Diferencias en el sistema de tipos, precisión, unidad, etc.**

EBD2014\_22 - Mg. Mercedes Vitturini

## Procesamiento de consultas

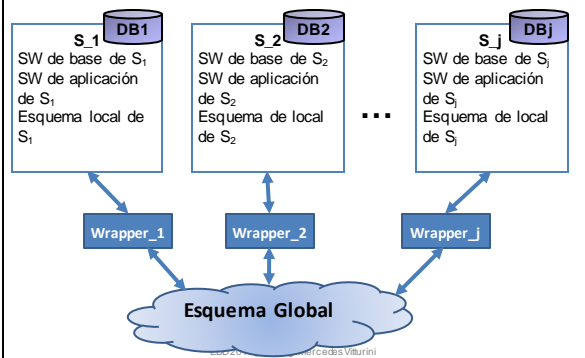
- La consulta se realiza sobre el esquema global, se traslada sobre los esquemas locales, se ejecuta. Los resultados se trasladan deben ser trasladados sobre el esquema global.
  - Cada sitio define su **wrapper:** librería que provee el mapeo del esquema local al esquema global y las traducciones de consultas globales a consultas locales.

Además:

- Reunir los resultados de consultas distribuidas requiere "limpieza de datos" (duplicados).
- Los sitios son autónomos y no siempre están dispuestos a compartir toda la información.

EBD2014\_22 - Mg. Mercedes Vitturini

## BDD Heterogéneas



EBD2014\_22 - Mg. Mercedes Vitturini

## Ontologías

- En informática se define como **ontología** a la definición de un esquema conceptual riguroso para un dominio acordado por una comunidad de expertos.
- La ontología facilita la comunicación y el intercambio de información entre diversos sistemas.
- Se asocia con el sentido semántico y la representación de conocimiento.

EBD2014\_22 - Mg. Mercedes Vitturini

## Conceptos estudiados

SMBDD Homogéneas

- Transacción Local
- Transacción global
- Atomicidad y compromiso de dos fases en SMBDD
- El coordinador de transacciones.
- Protocolos –
  - De compromiso distribuido
  - De compromiso de dos fases
  - Relación con el sistema de recuperación
- Gestión de estampillas de tiempo en sistemas distribuidos

- Gestión de deadlock en SMBDD
  - Prevención o detección
  - Protocolos centralizados y distribuidos

Administración de copias master-slave.

SMBDD Heterogéneas

- Características, ventajas y problemas.
- Consideraciones:
  - Vista unificada de los datos.
  - Procesamiento distribuido de consultas.
- Wrappers
- Ontologías

EBD2014\_22 - Mg. Mercedes Vitturini

## **Temas de la clase de hoy**

- **Bibliografía**

- “Principles of Database and Knowledge-Base Systems” – J. Ullman. Capítulo 10.
- “Database Systems Concepts” – A. Silberschatz. Capítulo 22 (edición 2005) y 19 (edición 2010).

EBD2014\_22 - Mg. Mercedes Vitturini