

## 18 BASES DE DATOS DISTRIBUIDAS

- Consisten en emplazamientos poco acoplados que no comparten componentes físicos
- Cada emplazamiento puede participar en la ejecución de transacciones que tienen acceso a los datos en uno o varios emplazamientos
- Los datos residen en varias ubicaciones
- Sistema con múltiples bd → software que permite que una colección heterogénea de bd se trate como una bd distribuida homogénea.

### 18.1 ALMACENAMIENTO DISTRIBUIDO DE DATOS

Una relación R puede almacenarse de 3 formas:

Sept 00: Problema 3

#### 18.1.1 Réplicas de los datos

Se guardan una o más réplicas (copias) idénticas de la relación R en emplazamientos distintos. Es completa si se guarda una copia en cada emplazamiento.

- **Ventajas:**
  1. **Disponibilidad** → Se puede ir procesando las consultas que impliquen a R aunque falle un emplazamiento
  2. **Aumento de paralelismo** → varios emplazamientos pueden procesar en paralelo consultas que impliquen la lectura de R, minimizando el tráfico de datos entre los emplazamientos
- **Inconvenientes:**
  1. **Aumento de la sobrecarga en las actualizaciones** → al tener que actualizarse todas las réplicas

Mejora el rendimiento de las ops de lectura, aumenta la disponibilidad para T's de sólo lectura pero las actualizaciones suponen una carga mayor.

Se puede simplificar la administración escogiendo una copia como *copia principal*.

#### 18.1.2 Fragmentación de datos

La relación R se divide en varios fragmentos, guardándose cada uno en un emplazamiento distinto. Puede ser:

##### 18.1.2.1 Fragmentación horizontal (filas)

- Un fragmento puede definirse como una selección ( $\sigma$ ) de la relación R, mediante un predicado P
- Se reconstruye R tomando la unión ( $\cup$ ) de todos los fragmentos

##### 18.1.2.2 Fragmentación vertical (columnas)

- Cada fragmento puede definirse como un subconjunto de atributos ( $\Pi$ ) de R
- Se reconstruye R tomando la reunión natural de todos los fragmentos

##### 18.1.2.3 Fragmentación mixta

Combinación anteriores

#### 18.1.3 Réplica y fragmentación de datos

Se puede aplicar la réplica y fragmentación de datos de forma sucesiva a la misma relación

## 18.2 TRANSPARENCIA DE LA RED

Se define como “el grado en que los usuarios del sistema pueden ignorar los detalles de la manera y del lugar en que se guardan los elementos de datos en los sistemas distribuidos”.

Se puede considerar desde varios aspectos:

### 18.2.1 Denominación de los elementos de datos

- Los elementos de datos deben tener nombres únicos. Soluciones:
  1. Exigir a todos los nombres que se registren en un servidor de nombres central.
    - Inconvenientes:
      - Este servidor se convierte en cuello de botella
      - Si se avería el servidor de nombres, puede que no sea posible que sigan funcionando todos los emplazamientos.
  2. Exigir que cada emplazamiento añada como prefijo su propio identificador de emplazamiento a todos los nombres que genere.
    - Ventajas:
      - No hay dos emplazamientos que generen el mismo nombre
      - No se necesita ningún control central.
    - Inconvenientes:
      - No se consigue la transparencia de la red dado que los identificadores de los emplazamientos están ligados a los nombres. Solución: alias (nombres sencillos que el sistema traduce en nombres reales) La correspondencia entre los alias y los nombres reales pueden guardarse en cada emplazamiento.
- Cada réplica y cada fragmento de un elemento de datos debe tener también nombre único.

### 18.2.2 Transparencia y actualizaciones

Es más difícil proporcionar transparencia en las actualizaciones, al tener que actualizar todas las réplicas.

## 18.3 PROCESAMIENTO DISTRIBUIDO DE CONSULTAS

### 18.3.1 Transformación de consultas

### 18.3.2 Procesamiento de reuniones sencillas

### 18.3.3 Estrategia de semirreunión

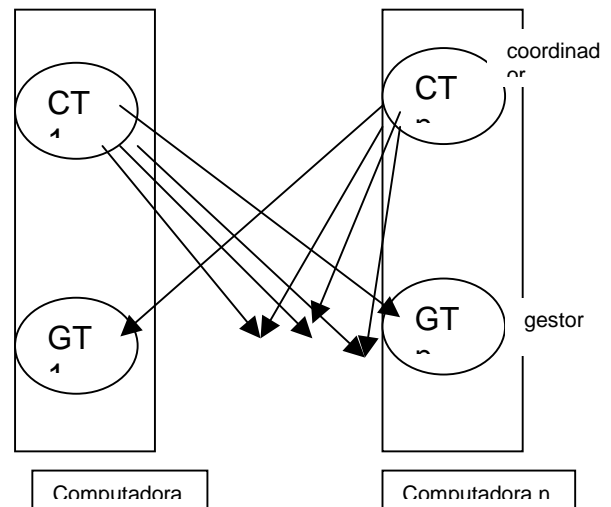
### 18.3.4 Estrategia de reunión para explotar el paralelismo

## 18.4 MODELO DE TRANSACCIONES DISTRIBUIDAS

### 18.4.1 Arquitectura del sistema

Cada emplazamiento del sistema consta de 2 subsistemas:

- **Gestor de transacciones**
  - Conservar un registro histórico con fines de recuperación
  - Participar en un esquema adecuado de control de concurrencia para coordinar la ejecución concurrente
- **Coordinador de transacciones**
  - Iniciar la ejecución de la transacción
  - Dividir la transacción y distribuirlas para su ejecución en los emplazamientos
  - Coordinar la terminación (comprometer o abortar)



### 18.4.2 Modos de fallo del sistema

- Fallo emplazamiento
- Pérdida mensajes → se utilizan protocolos de control, como TCP/IP para tratar estos errores
- Fallo de un enlace de comunicaciones
- División de la red

Los efectos de estos dos fallos dependen del modo en que están interconectados los emplazamientos

Cada configuración presenta ventajas e inconvenientes. Para compararlas hay que fijarse en:

1. Coste de instalación (enlazar físicamente)
2. Coste comunicaciones (tiempo y dinero)
3. Disponibilidad (a pesar fallos)

Redes conectadas completamente	Coste de instalación desorbitado al crecer el nº enlaces
Redes parcialmente conectadas	Mayor coste de comunicación al tener que encaminar los mensajes
Red en árbol	Coste de instalación y comunicación bajo El fallo de un solo enlace puede hacer que la red quede dividida
Red en estrella	El fallo de un solo enlace da lugar a la división de la red Bajo coste comunicación El fallo del enlace central puede dar lugar a que todos queden desconectados
Red en anillo	Grado de disponibilidad mayor que las de árbol Coste comunicación elevado

### 18.4.3 Robustez

→ Es la Capacidad de detectar los fallos, volver a configurar el sistema y recuperarse cuando se repare.

→ Tratamiento según tipo de fallo:

- ♦ Pérdida de mensajes: Se trata mediante la retransmisión
- ♦ Fallo de emplazamiento y división de red: No es fácil diferenciarlos.

→ Se debe evitar:

- ♦ . Que se elijan dos o mas servidores centrales en particiones disjuntas
- ♦ . Que más de una partición actualice un elemento de datos replicado

→ La reincorporación al sistema de un emplazamiento o de un enlace reparado requiere tomar precauciones para actualizar sus tablas.

## 18.5 PROTOCOLO DE COMPROMISO

→ Se utiliza para asegurar la **atomicidad** : O T se compromete en todos los emplazamientos o aborta en todos ellos.

### 18.5.1 Compromiso de dos fases (C2F)

T: Transacción iniciada en el emplazamiento Ei

Ci: Coordinador de transacciones en Ei

Gi: Gestor de transacciones de Ei

Para entenderlo mejor, problemas de examen resueltos...>21 y 22

#### 18.5.1.1 El protocolo de compromiso

Cuando T completa su ejecución, Ci inicia C2F

##### ▪ Fase 1:

Ci hace :

- 1.- Añade el registro <T **preparar**> al registro histórico y se guarda en un almacenamiento estable.
- 2.- Envía mensaje **preparar** T a todos los emplazamientos en que se ejecutó T

Gi recibe el mensaje y **decide si compromete** la parte de T

**NO** → Añade un registro <**no** T> al registro histórico

Envía un mensaje **abortar** T a Ci

**SI** → Añade un registro <T **preparada**> al registro histórico

Obliga a guardar el registro histórico en un almacenamiento estable

Envía un mensaje **preparada**

##### ▪ Fase 2:

Cuando Ci recibe respuestas de todos los emplazamientos al mensaje preparar T o si transcurre un intervalo de tiempo predeterminado, Ci determina si una transacción T se puede

**comprometer** → Si Ci recibe un mensaje T **preparada** de **todos** los emplazamientos ⇒

Añade al registro histórico <T **comprometida**>

Obliga a guardar el registro histórico en un almacenamiento estable.

(El destino de la transacción queda sellado)

Envía un mensaje **comprometer** T

**abortar** → Si Ci **NO** recibe un mensaje T **preparada** de **todos** los emplazamientos ⇒

Añade al registro histórico <T **abortada**>

Obliga a guardar el registro histórico en un almacenamiento estable.

(El destino de la transacción queda sellado)

Envía un mensaje **abortar** T

### 18.5.1.2 Tratamiento de fallos del protocolo C2F

#### ▪ Fallo de un emplazamiento participante

- ♦ Si  $C_i$  detecta que ha fallado un emplazamiento  $E_k$  hace:

Caso	Solucion
$E_k$ falla <b>antes</b> de responder a $C_i$ con mensaje <b>T preparada</b>	se da por supuesto que ha repondido con un mensaje <b>abortar T</b>
$E_k$ falla <b>despues</b> de responder a $C_i$ con mensaje <b>T preparada</b>	se ejecuta el resto del protocolo de compromiso ignorando el fallo.

- ♦ Si  $E_k$  se recupera de un fallo, debe examinar el registro histórico para determinar el destino de las transacciones que estaban en trance de ejecución cuando tuvo lugar el fallo. Según lo que contenga este registro se considerarán diferentes casos:

El registro histórico contiene:	El emplazamiento $E_k$ hace:
<T <b>comprometida</b> >	Ejecuta <b>rehacer</b> (T)
<T <b>abortada</b> >	Ejecuta <b>deshacer</b> (T)
<T <b>preparada</b> >	<p>Consulta a <b><math>C_i</math></b> el destino de T:</p> <ul style="list-style-type: none"> <li>• Si <b><math>C_i</math></b> funciona, notificará a <math>E_k</math> si T se debe comprometer → <math>E_k</math> Ejecuta <b>rehacer</b> (T) o abortar → <math>E_k</math> Ejecuta <b>deshacer</b> (T)</li> <li>• Si <b><math>C_i</math></b> falla, <math>E_k</math> se busca la vida:  <math>E_k</math> envia un mensaje <b>consulta-estado</b> a todos los E.            Los E, consultan sus registros históricos e informan a <math>E_k</math> si T se ha ejecutado allí y si ha sido comprometido o abortado.            Si ninguno contesta, se aplaza la decisión  <math>E_k</math> envia periódicamente el mensaje <b>consulta-estado</b> hasta que algún E le responda</li> </ul>
No contiene ningún registro de control referido a T <b>Comprometida, abortada, preparada</b>	<p>Significa que <math>E_k</math> falló antes de responder a <b><math>C_i</math></b>, luego según el algoritmo debe abortar T. Por tanto:</p> <p>Ejecuta <b>deshacer</b> (T)</p>

#### ▪ Fallo del coordinador

Si el coordinador,  **$C_i$** , falla en medio de la ejecución del protocolo de compromiso de la transacción T, los emplazamientos participantes deben decidir el destino de T.

Ek activo contiene en su registro histórico: <T <b>comprometida</b> > <T <b>abortada</b> >	T se debe Comprometer Abortar
Ek activo NO contiene en su registro histórico: <T <b>preparada</b> >	Es preferible abortar T
Todos los E activos contienen sólo <T <b>preparada</b> >	T queda bloqueado hasta que se recupere <b><math>C_i</math></b> <b>-ES EL PROBLEMA MAS GRAVE-</b>

### ▪ División de la red

Ci y todos E participantes quedan en una partición	El fallo no tiene ningún efecto sobre <b>C2F</b>
Ci y todos E participantes pertenecen a varias particiones diferentes	

#### 18.5.1.3 Recuperaciones y control de la concurrencia

Son transacciones **dudosas** aquella en que se encuentra un registro <T **preparada**> en el registro histórico, pero no se encuentra ni un registro <T **comprometida**> ni uno <T **abortada**>. Como puede ser un proceso lento averiguar el estado de estas transacciones, en lugar de escribir en el registro histórico un registro <T **preparada**>, se escribe un registro <T L **preparada**>, donde L es una lista de todos los bloqueos mantenidos por la transacción T en el momento de escribirlo. En el momento de la recuperación, después del desarrollo de las acciones locales de recuperación, se reclaman todos los bloqueos anotados en el registro histórico en los registros <T L **preparada**> para cada transacción. Así, el compromiso o retroceso de las transacciones dudosas se lleva a cabo de manera concurrente con la ejecución de las nuevas transacciones.

#### 18.5.2 Compromiso de tres fases (C3F)

Está diseñado para evitar la posibilidad de bloqueo en un caso restringido de posibles fallos. Exige que:

- No pueden producirse divisiones de red
- Como máximo k ( parámetro de tolerancia a fallos) emplazamientos participantes pueden fallar mientras se ejecute el protocolo C3F
- En cualquier momento deben funcionar k+1 emplazamientos

Este protocolo obtiene la propiedad de no bloqueo, añadiendo una fase mas en la que se toma una decisión previa sobre el destino de T

##### 18.5.2.1 El protocolo de compromiso

No hay problemas de este protocolo

#### ▪ Fase 1: =Fase 1 del protocolo C2f

#### ▪ Fase 2:

La diferencia básica con el protocolo 2CF es que se añade **precomprometer** y **acuse-recibo**

#### ▪ Fase 3:

Sólo se ejecuta si la decisión de la fase 2 fue **precomprometer**

El coordinador debe esperar hasta que recibe por lo menos K mensajes **acuse-recibo** Entonces añade al registro histórico <T **comprometida**> guardándolo en un almacenamiento estable

Ci envia mensaje **comprometer** T a todos los emplazamientos y éstos guardan la información en su registro histórico

### 18.5.2.2 Manejo de fallos

### 18.5.2.3 Protocolo de fallo del coordinador

Pienso que si no sale lo anterior en exámen, esto mucho menos

### 18.5.3 Comparación de protocolos

- El protocolo C2F se utiliza mucho, a pesar de su posibilidad de bloqueo. La probabilidad de que se produzca un bloqueo es bastante baja como para que se justifique el coste adicional del protocolo C3F
- C3F tiene mas vulnerabilidad a los fallos

## 18.6 SELECCIÓN DEL COORDINADOR

Si el coordinador falla debido al fallo del emplazamiento en que reside, el sistema sólo puede continuar la ejecución reiniciando un nuevo coordinador en otro emplazamiento.

- Puede conservarse una copia de seguridad del coordinador, o bien
- Puede elegirse un nuevo coordinador → algoritmo de elección

### 18.6.1 Coordinador copia de seguridad

- Es un emplazamiento que, además de otras tareas, conserva localmente suficiente información como para reemplazar al coordinador en un fallo.
- Todos los mensajes dirigidos al coordinador los reciben tanto él, como su copia de seguridad
- El coordinador copia de seguridad ejecuta los mismos algoritmos que el original
- La única **diferencia** de funciones es que el de seguridad, no emprende ninguna acción que pueda afectar a otros emplazamientos.
- La **ventaja** principal → posibilidad de continuar inmediatamente el procesamiento
- El **inconveniente** → sobrecarga de la ejecución duplicada de las tareas del coordinador y a necesidad de tener que comunicarse entre ellos a intervalos regulares

### 18.6.2 Algoritmos de elección

- Se asocia un número de identificación único ( $i$ ) a cada emplazamiento ( $E_i$ ) activo del sistema.
- El coordinador reside siempre en el emplazamiento con el número de identificación más elevado.
- El objetivo de los algoritmos de elección es escoger un nuevo emplazamiento para el coordinador (es decir, el del número más elevado).
- Algoritmo **luchador**:  
Consiste en que si un emplazamiento no recibe respuesta del coordinador, supone que ha fallado. Entonces se erige él mismo, nuevo coordinador, y envía mensajes al resto de emplazamientos para comunicarlo. Si uno de ellos contesta ¡He, que yo soy mayor!, se queda como coordinador, y hace lo mismo otra vez. Así hasta que ninguno contesta, entonces el último en enviar el mensaje, se supone que es el mayor, y se queda como coordinador

## 18.7 CONTROL DE LA CONCURRENCIA

Se supone que cada emplazamiento participa en la ejecución de un protocolo de compromiso para asegurar la atomicidad de la transacción global



### 18.7.1 Protocolos de bloqueo

La única modificación respecto a los descritos en el capítulo 14 es la manera en que se implementa el gestor de bloqueos, que puede ser:

#### 18.7.1.1 Enfoque de gestor de bloqueo único

- Único gestor de bloqueo, que reside en un único emplazamiento seleccionado ( $E_i$ )
- Todas las transacciones de bloqueo y desbloqueo pasan por  $E_i$ 
  - **Ventajas:**
    - Implementación sencilla. (2 mensajes: uno para tratar las solicitudes de bloqueo, otro para tratar las de desbloqueo)
    - Tratamiento sencillo de interbloqueos: (se pueden aplicar directamente las técnicas del capítulo 14)
  - **Inconvenientes:**
    - Cuello de botella: (todas las peticiones tienen que pasar por  $E_i$ )
    - Vulnerabilidad: (Si falla  $E_i$ , se pierde el control de concurrencia)

#### 18.7.1.2 Varios coordinadores

- La función del gestor de bloqueo está distribuida entre varios emplazamientos)
- Cada gestor de bloqueo administra las solicitudes de bloqueo y desbloqueo de un subconjunto de elementos dados.
- Cada gestor de bloqueos reside en un emplazamiento diferente
  - **Ventajas**
    - Disminuye cuello botella

#### 18.7.1.3 Protocolo de mayoría

- Cada emplazamiento conserva un gestor local de bloqueo
- Este gestor local de bloqueo gestiona las solicitudes de bloqueo y desbloqueo de los elementos de datos guardados en ese emplazamiento
- Si el elemento de datos  $Q$  **no está replicado** y reside en el emplazamiento  $E_i$ :
  - Se envía un mensaje al gestor de bloqueos de  $E_i$
  - Si no puede concederla, se espera hasta que pueda
  - Si se puede conceder, el gestor de bloqueos de  $E_i$  devuelve el mensaje al solicitante para indicarle que ha concedido la solicitud de bloqueo
  - **Ventajas** → Sencillez implementación (un mensaje bloqueo, otro desbloqueo)
  - **Inconvenientes** → Tratamiento de los interbloqueos complicado
- Si el elemento de datos  $Q$  **está replicado** en  $n$  emplazamientos
  - Hay que enviar mensaje de solicitud de bloqueo a mas de la mitad de los mismos
  - Cada gestor determina si se puede conceder el bloqueo de forma inmediata
  - La respuesta queda pospuesta hasta que se pueda conceder la solicitud.
  - La transacción no opera sobre  $Q$  hasta que ha logrado obtener bloqueos sobre la mayor parte de las réplicas de  $Q$
  - **Ventajas** → Trata los datos replicados de forma descentralizada
  - **Inconvenientes:**
    - Implementación: Se necesitan  $2(n/2+1)$  mensajes para las solicitudes de bloqueos y  $(n/2 + 1)$  para las de desbloqueo
    - Tratamiento de los interbloqueos: al no realizarse ambas operaciones en un único emplazamiento.



#### 18.7.1.4 Protocolo sesgado

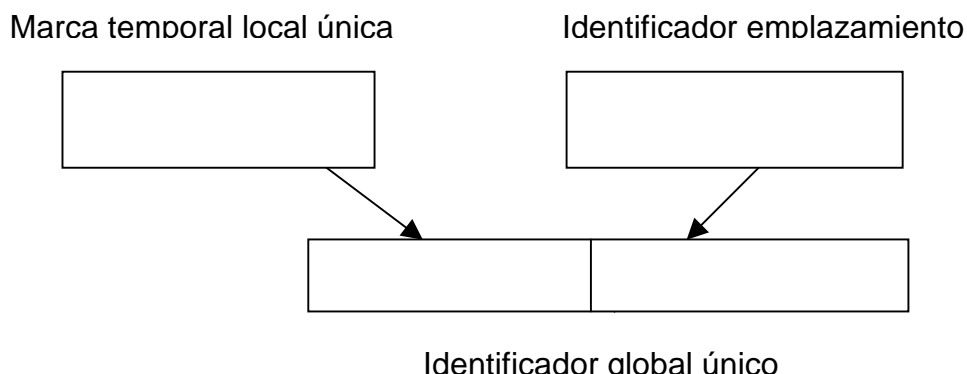
- El sistema conserva un gestor de bloqueo en cada emplazamiento.
- Cada administrador gestiona los bloqueos de todos los elementos de datos guardados en ese emplazamiento
- Se diferencia el protocolo de mayoría en que las solicitudes de bloqueos compartidos reciben un tratamiento más favorable que las de los bloqueos exclusivos
  - **Bloqueos compartidos** : Se solicita un bloqueo sobre Q al gestor de bloqueo de **UN** emplazamiento que contenga una réplica de Q:
  - **Bloqueos exclusivos**: Se solicita un bloqueo sobre Q al gestor de bloqueo de **TODOS** los emplazamientos que contengan réplicas de Q:
- La respuesta de solicitud se pospone hasta que se pueda conceder.
  - **Ventajas**: Impone **menos sobrecarga** en las operaciones **leer** que el protocolo de mayoría.
  - **Inconvenientes**: **Sobrecarga** añadida en **escribir**

#### 18.7.1.5 Copia principal

- Se selecciona una réplica como copia principal
- Para cada elemento de datos Q, la copia principal de Q debe residir exactamente en el emplazamiento principal de Q
- Cuando se necesita bloquear un elemento de datos Q se solicita un bloqueo al emplazamiento principal de Q
- Si no se puede conceder, se pospone
  - **Ventajas**:
    - La copia principal permite que el control de la concurrencia para los datos replicados se trate igual que para los no replicados, lo que permite una
    - Implementación sencilla
  - **Inconvenientes**:
    - Si el emplazamiento principal de Q falla, Q se vuelve inaccesible

#### 18.7.2 Marcas temporales

- Se da a cada transacción una **marca temporal única** que el sistema utiliza para decidir el orden de secuenciación



- 2 métodos principales para generar marcas:
  - **centralizado**: Se escoge un único emplazamiento para generar las marcas
  - **distribuido**:

- Cada emplazamiento genera una marca temporal única mediante un contador lógico o reloj local (RLi)
- La marca temporal global única se obtiene concatenando la marca temporal local única con el identificador del emplazamiento, que también debe ser único
- El orden de concatenación es importante. Se utiliza el identificador del emplazamiento en la posición menos significativa para asegurarse de que las marcas temporales globales generadas en un emplazamiento no sean siempre mayores que las generadas en otro.
- En cada emplazamiento  $E_i$  se define un reloj lógico (RLi), que genera la marca temporal única
- Este RLi puede construirse con un contador que se incrementa después de generarse cada nueva marca temporal local
- Para asegurarse de que están sincronizados, hay que exigir que el emplazamiento  $E_i$  adelante su reloj lógico siempre que una transacción  $T_i$  con la marca temporal  $\langle x, y \rangle$  visite ese emplazamiento y  $x > \text{valor actual de RLi} \rightarrow E_i$  adelanta su reloj lógico a  $x+1$

## 18.8 TRATAMIENTO DE LOS INTERBLOQUEOS

- Si se permiten que se produzcan interbloqueos y se confía en su detección, el problema principal en sistemas distribuidos radica en decidir la manera de conservar el grafo de espera.
- Cada emplazamiento conserva un grafo local de espera
- Si cualquier grafo local de espera tiene un ciclo  $\rightarrow$  se habrá producido interbloqueo
- Si no hay ciclos en ninguno de los grafos de espera, no significa que no haya interbloqueos

### 18.8.1 Enfoque centralizado

- Se genera y se conserva en un **único** emplazamiento un grafo global de espera (unión de todos los grafos locales)  $\rightarrow$  **coordinador de detección de interbloqueos**
- 2 tipos de grafos de espera:
  - grafo real  $\rightarrow$  describe el estado real pero desconocido del sistema en cada momento
  - grafo construido  $\rightarrow$  es una aproximación generada por el controlador durante la ejecución del algoritmo del controlador de tal forma que siempre que se invoque el algoritmo de detección, los resultados comunicados sean correctos, es decir, **si existe un bloqueo se comunique con rapidez, y que efectivamente sea un interbloqueo.**
- Cuando se invoca el algoritmo de detección de interbloqueos,
  - el coordinador busca en su grafo local. Si se halla un ciclo, se selecciona una víctima para hacerlo retroceder.
  - El coordinador debe notificar a todos los emplazamientos que se ha seleccionado como víctima una transacción determinada.
  - Los emplazamientos, a su vez, hacen retroceder a la transacción víctima
- Este esquema puede producir retrocesos innecesarios como resultado de una de las situaciones siguientes:
  - Ciclos falsos en el grafo global de espera

- Se haya producido realmente interbloqueo y se haya escogido una víctima, mientras que una de las transacciones se abortó por razones no relacionadas con el interbloqueo.

### 18.8.2 Enfoque completamente distribuido

Está muy bien explicado en problemas de examen:  
Apuntes: Problema 20

- Los controladores comparten por igual la responsabilidad
- Una transacción se ejecuta en un solo emplazamiento y realiza peticiones a otros emplazamientos para acceder a los datos no locales
- El grafo total está compuesto por una serie de grafos locales, uno para cada uno de los emplazamientos que tenga nuestro SGBD distribuido
- Cada emplazamiento conserva su propio grafo local de espera tal que si  $T_i \rightarrow T_k$  significa que  $T_i$  está esperando un recurso que posee  $T_k$  sea o no de su propio emplazamiento  
 $T_{ex}$  es un emplazamiento adicional tal que  
 $T_{ex} \rightarrow T_i$  significa que otro emplazamiento está esperando un recurso que posee la transacción  $T_i$   
 $T_i \rightarrow T_{ex}$  significa que  $T_i$  está esperando un recurso situado en otro emplazamiento
- Si un grafo local de espera contiene **un ciclo que no** implica a  $T_{ex} \rightarrow$  el sistema **se halla** en estado de interbloqueo
- Si un grafo local de espera contiene **un ciclo que si** implica a  $T_{ex} \rightarrow$  cabe la **posibilidad** de interbloqueo, entonces para saber si existe o no hay que invocar un algoritmo distribuido de detección de interbloqueos:
  - Se asigna a cada transacción  $T_i$  un identificador único que se denota  $ID(T_i)$
  - Cuando el emplazamiento  $E_k$  descubre que su grafo local de espera contiene un ciclo que implica a  $T_{ex}$  de la forma  $T_{ex} \rightarrow T_{k1} \rightarrow T_{k2} \rightarrow \dots T_{kn} \rightarrow T_{ex}$   
 Sólo enviará un mensaje de detección de interbloqueos a otro emplazamiento si  $ID(T_{kn}) < ID(T_{ki})$ . ( En caso contrario el emplazamiento  $E_k$  continúa con su ejecución normal)
  - El emplazamiento  $E_k$ , a su vez, repite el procedimiento anterior. Así, después de un número finito de pasadas, si existe un interbloqueo, se descubre; si no existe ningún interbloqueo, el proceso de detección de interbloqueos se detiene

## 18.9 SISTEMAS CON MULTIPLES BASES DE DATOS

La integración completa de los sistemas existentes en una base de datos distribuida homogénea resulta con frecuencia difícil o imposible debido a:

- **Dificultades técnicas:** Costes, etc
- **Dificultades de las organizaciones:** políticamente imposible

Los sistemas con múltiples bases de datos proporcionan un entorno en el que las nuevas aplicaciones de bases de datos pueden tener acceso a los datos de gran variedad de bases de datos existentes ubicadas en diferentes entornos de hardware y de software heterogéneos

- Crean la ilusión de la integración lógica de bases de datos sin exigir su integración física

- Tienen que permitir a los sistemas locales de bases de datos conservar un elevado grado de autonomía respecto a la base de datos local y a las transacciones que se ejecuten con esos datos

### 18.9.1 Visión unificada de los datos

Para crear la ilusión de un sistema de bases de datos único e integrado hay que

- **utilizar un modelo de datos común.** La elección natural es el modelo relacional, con SQL como lenguaje de consulta común.
- **Proporcionar un esquema conceptual común**→Es una tarea muy dificultosa. La solución habitual es confiar sólo en la optimización en el nivel local

### 18.9.2 Gestión de las transacciones

- Los sistemas con múltiples bases de datos permiten dos tipos de transacciones:
  - 1.- Transacciones **locales**→las ejecuta cada SGBD local fuera del control del SMDB
  - 2.- Transacciones **globales**→se ejecutan bajo el control del SMDB
- El SMDB es consciente del hecho de que pueden ejecutarse transacciones locales en los emplazamiento locales, pero no de las transacciones concretas que se están ejecutando ni de los datos a los que puedan tener acceso
- Asegurar la autonomía local de cada SGBD exige no realizar ninguna modificación en el software del SGBD local→ el SGBD de un emplazamiento no puede comunicarse directamente con el de ningún otro emplazamiento para sincronizar la ejecución de una transacción global activa en varios emplazamientos
- Cada SGBD debe utilizar un esquema de control de concurrencia para asegurar que su planificación sea secuencial

#### 18.9.2.1 Secuencialidad de dos niveles (S2N)

Asegura la secuencialidad en los dos niveles del sistema

- Cada GBD local asegura la secuencialidad local entre sus transacciones locales, incluidos los que forman parte de transacciones globales
- El SMDB sólo asegura la secuencialidad entre las transacciones globales, **ignorando el orden inducido por las transacciones locales**
- **El protocolo de lectura global** permite que las transacciones globales lean, pero no que actualicen, los elementos locales de datos, mientras que impiden cualquier acceso a los datos globales por parte de las transacciones locales
- **El protocolo de lectura local** concede a las transacciones locales acceso para lectura a los datos globales, pero impiden cualquier acceso a los datos locales por parte de las transacciones globales. Una transacción tiene **dependencia de valores** si el valor que escribe en un elemento de datos de un emplazamiento depende del valor que lee en un elemento de datos de otro emplazamiento
- **El protocolo de lectura y escritura global y lectura local** es el más generoso . Permite que las transacciones globales lean y escriban los datos locales y que las transacciones locales lean lo datos globales.

#### 18.9.2.2 Aseguramiento de la secuencialidad global

- En entornos en que se puedan ejecutar actualizaciones de transacciones de **sólo lectura**:  
 Se crea en cada SGBD local un elemento de datos especial denominado **billete**.  
 Cada transacción global que tiene acceso a los datos de un emplazamiento debe escribir el billete de ese emplazamiento:

- asegura que las transacciones globales entren en conflicto directamente en cada emplazamiento que visiten
- El gestor de transacciones globales puede controlar el orden en el que se serializan las transacciones globales controlando el orden en el que se tiene acceso a los billetes
- En entornos en que **no se generan conflictos locales directos** en cada emplazamiento, si se supone que las planificaciones locales son de tal modo que el orden de compromiso y el de secuencialidad son siempre idénticos,
  - Se puede asegurar la secuencialidad controlando sólo el orden en que se comprometen las transacciones
- El **problema** con las planificaciones que aseguren la secuencialidad global es que pueden restringir excesivamente la concurrencia