CONCURRENCIA en BASES DE DATOS

- Clasificación de los SISTEMAS DE BASES DE DATOS según el número de usuarios que pueden utilizarlos de forma CONCURRENTE (al mismo tiempo)
  - MONOUSUARIO
  - MULTIUSUARIO

- El que varios usuarios puedan usar un mismo computador a la vez, se debe a la MULTIPROGRAMACIÓN: el computador puede procesar al mismo tiempo varios programas (transacciones)
  - Si el equipo tiene **varias CPU**, es posible el PROCESAMIENTO SIMULTÁNEO de transacciones
  - Si **CPU única**, el **SO de Multiprogramación** reparte el tiempo de CPU entre los programas: EJECUCIÓN CONCURRENTE INTERCALADA (*modelo que supondremos*)

  figura 1

- Las transacciones introducidas por los usuarios, que se ejecutan de manera concurrente, pueden leer/actualizar los MISMOS elementos de la BD

¿Por qué es necesario el CONTROL de la CONCURRENCIA?

- ... porque pueden surgir PROBLEMAS cuando las transacciones concurrentes se ejecutan de manera NO controlada

- Ejemplo de la base de datos para hacer reservas de vuelos
  - Se almacena un registro por cada vuelo
  - Cada registro incluye (entre otras cosas) el nº de asientos reservados en el vuelo
  - Dos transacciones concurrentes:
    - **T1** cancela N reservas en un vuelo V_X y reserva N plazas en otro V_Y
    - **T2** reserva M asientos en el vuelo V_X

  figura 2
  - Un programa para esta BD tiene como parámetros
    - Los números de vuelos,
    - sus fechas,
    - el número de plazas que reservar/cancelar
  - Así que el mismo programa puede usarse para ejecutar muchas transacciones (NOTA: una transacción es la EJECUCIÓN ESPECÍFICA de un programa)

Problemas

- EL PROBLEMA DE LA ACTUALIZACIÓN PERDIDA
  - Dos transacciones T1 y T2 que acceden a los mismos datos, tienen sus operaciones intercaladas de modo que hacen incorrecto el valor de algún dato
- EL PROBLEMA DE LA ACTUALIZACIÓN TEMPORAL (o LECTURA SUCIA)
  - Una transacción T1 actualiza un elemento X de la BD y luego falla, pero otra transacción T2 tiene acceso a X ANTES de que se restaure el valor original de X
- EL PROBLEMA DEL RESUMEN INCORRECTO
  - Una transacción T1 calcula una función agregada de resumen sobre varios registros, mientras otras transacciones actualizan dichos registros: puede que T1 considere algunos registros antes de que se actualicen y otros después de actualizarse.
- EL PROBLEMA DE LA LECTURA NO REPETIBLE
  - Una transacción T1 lee un elemento X dos veces y otra transacción T2 modifica dicho X entre las dos lecturas: T1 recibe diferentes valores para el mismo elemento

figura 3

Planes y Recuperabilidad

» PLAN: orden de ejecución de las operaciones de varias transacciones que se ejecutan de manera concurrente e intercalada

PLAN o HISTORIA **de TRANSACCIONES**

- Un plan P de n transacciones T_1, T_2, ..., T_n es un
  ordenamiento para las operaciones de las transacciones,
  sujeto a la restricción de que
  para cada transacción T_i que participa en P,
  las operaciones de T_i deben aparecer en el mismo orden
  en que ocurren en T_i
- En P, las operaciones de otras T_k pueden intercalarse con las de T_i

figura 4

Planes y Recuperabilidad

- Es posible caracterizar los tipos de planes que facilitan la recuperación tras un fallo
- Para fines de control de concurrencia (y para recuperación), interesan las operaciones de
  - LEER_ELEMENTO (l),
  - ESCRIBIR_ELEMENTO (e),
  - COMMIT (c) y
  - ROLLBACK (r)

P_a: $l_1(X)$; $l_2(X)$; $e_1(X)$; $l_1(Y)$; $e_2(X)$; c_2 ; $e_1(Y)$; c_1 ;

P_b: $l_1(X)$; $e_1(X)$; $l_2(X)$; $e_2(X)$; c_2 ; $l_1(Y)$; r_1 ;

Planes y Recuperabilidad

- Dos OPERACIONES de un plan P están EN CONFLICTO si
  - pertenecen a diferentes transacciones,
  - tienen acceso al mismo elemento X,
  - y una de las dos operaciones es ESCRIBIR_ELEMENTO(X)

ejemplo: operaciones en conflicto en P_a:

$\mathbf{l_1(X)}$; $l_2(X)$; $e_1(X)$; $l_1(Y)$; $\mathbf{e_2(X)}$; c_2 ; $e_1(Y)$; c_1;
$l_1(X)$; $\mathbf{l_2(X)}$; $\mathbf{e_1(X)}$; $l_1(Y)$; $e_2(X)$; c_2 ; $e_1(Y)$; c_1;
$l_1(X)$; $l_2(X)$; $\mathbf{e_1(X)}$; $l_1(Y)$; $\mathbf{e_2(X)}$; c_2 ; $e_1(Y)$; c_1 ;

ejemplo: operaciones NO en conflicto en P_a:

$\mathbf{l_1(X)}$; $\mathbf{l_2(X)}$; $e_1(X)$; $l_1(Y)$; $e_2(X)$; c_2 ; $e_1(Y)$; c_1;
$l_1(X)$; $l_2(X)$; $e_1(X)$; $l_1(Y)$; $\mathbf{e_2(X)}$; c_2 ; $\mathbf{e_1(Y)}$; c_1;
etc.

Planes y Recuperabilidad

Un **plan P** es **COMPLETO** si ...

1. Las operaciones de P son las de T_1, T_2, ..., T_n incluida una última operación de COMMIT o ROLLBACK para cada transacción en el plan
2. Para cualquier par de operaciones en la misma transacción T_i su orden de aparición en P es el mismo que su orden de aparición en T_i
3. Para cualesquiera dos operaciones en conflicto, una de ellas debe ocurrir antes que la otra en el plan

- Estas condiciones permiten que dos operaciones que NO estén en conflicto, ocurran en P SIN DEFINIR CUÁL SE EJECUTA PRIMERO
- Por tanto, un plan es un ORDEN PARCIAL de operaciones de n transacciones
- Pero es necesario ESPECIFICAR un ORDEN TOTAL entre...
  - TODO PAR DE OPERACIONES EN CONFLICTO (3)
  - CUALQUIER PAR DE OPERACIONES DE LA MISMA TRANSACCIÓN T_i (2)
- Un plan completo NUNCA contiene transacciones activas
  (todas hacen COMMIT o ROLLBACK)

Planes con base en su recuperabilidad

- Hemos de asegurar que una vez que T se ha confirmado, nunca será necesario anularla (cancelarla, revertirla) --- **planes recuperables**

1. PLAN RECUPERABLE

Un **plan P** es **recuperable** si ninguna transacción T de P se confirma antes de haberse confirmado toda transacción T' que ha escrito un dato que T lee.

- Una transacción T lee de la transacción T' en un plan P, si T' escribe un elemento X y luego T lo lee
- T NO LEE de T'...
  - Si en P, T' ha abortado antes de que T lea el elemento X
  - Si otras transacciones escriben X después de que T' lo haya escrito y antes de que T lo lea

Ejemplo de plan no recuperable

$$P_c: l_1(X) ; e_1(X) ; l_2(X) ; l_1(Y) ; e_2(X) ; c_2 ; ...$$

SOLUCIÓN: postergar la confirmación de T_2 hasta que T_1 se confirme

$$P_d: l_1(X) ; e_1(X) ; l_2(X) ; l_1(Y) ; e_2(X) ; e_1(Y) ; c_1 ; c_2;$$

Planes con base en su recuperabilidad

- En un plan recuperable ninguna T **confirmada** tiene que anularse jamás,
- pero puede ocurrir el fenómeno de ANULACIÓN EN CASCADA
  - Una T **NO confirmada** debe anularse porque ha leído X de T' y T' ha sido abortada

$$P_e: l_1(X) ; e_1(X) ; l_2(X) ; l_1(Y) ; e_2(X) ; e_1(Y) ; r_1 ;$$

  - La cancelación en cascada consume mucho tiempo

2. PLAN SIN CANCELACIÓN EN CASCADA

Un **plan P evita la cancelación en cascada** si toda T en el plan sólo lee datos escritos por transacciones confirmadas

¿Cómo transformamos P_e para que evite la cancelación en cascada?

Planes con base en su recuperabilidad

3. PLAN ESTRICTO

Un **plan P** es **estricto** si las transacciones NO pueden leer ni escribir un elemento X hasta que sea confirmada o abortada toda transacción que haya escrito X

- Si T1 es abortada, **recuperar** (deshacer) **una** operación de **escritura** $e_1(X,5)$ consiste en **restaurar el valor anterior** del elemento X
- Pero esto puede no funcionar correctamente si el plan NO es estricto:

$$P_f: e_1(X,5) ; e_2(X,8) ; r_1 ;$$

- Un plan estricto...
  - No tiene este problema
  - Es recuperable y
  - Evita la cancelación en cascada

Seriabilidad de los planes

- Si NO SE PERMITE LA INTERCALACIÓN, sólo hay DOS formas de ordenar las operaciones de dos transacciones T1 y T2
  - Ejecutar T1 completa, seguida de T2 completa
  - Ejecutar T2 completa, seguida de T1 completa

figura 4(a) y (b)

- Si SE PERMITE LA INTERCALACIÓN DE OPERACIONES, existen muchos órdenes posibles

figura 4(c)

¿Cuáles son correctos?
¿existen técnicas de control de concurrencia
que eviten planes incorrectos?

TEORÍA de la SERIABILIDAD

Seriabilidad de los planes

PLAN EN SERIE

- Un **plan P** es **en serie** si, por cada transacción T que participa en el plan, todas las operaciones de T se ejecutan consecutivamente en el plan
- de lo contrario, P es un plan **no en serie**

  - Si las transacciones son independientes, todo plan en serie se considera correcto:
    - toda transacción es correcta si se ejecuta por sí sola (sin interferencias) y
    - las transacciones son independientes entre sí
    - ... O sea que no importa cuál se ejecute primero
  - Pero los planes en serie limitan la concurrencia $\Rightarrow$ en general, son inaceptables

- Hemos de determinar qué planes no en serie ...
  - *siempre* dan un resultado correcto y
  - cuáles *pueden* dar un resultado erróneo

Seriabilidad de los planes

PLAN SERIABLE

- Un **plan P** es **seriable** si es equivalente a algún plan en serie de las mismas n transacciones
- Un plan que no es equivalente a ningún plan en serie, es no seriable
- Si un plan no en serie P es seriable, entonces P es **correcto**

pero... ¿cuándo dos planes son equivalentes?

Pues pueden serlo ...

- POR RESULTADOS: si producen el mismo estado final en la base de datos
  - ¿Y si eso se cumple por casualidad? --- no sirve!

  figura 5, si X=100 (y además, son transacciones distintas!!)
- Si las operaciones que se aplican a cada dato afectado por los planes, se aplican al elemento en el mismo orden en ambos planes
  - **EQUIVALENCIA POR CONFLICTOS**
  - EQUIVALENCIA DE VISTAS (menos restrictiva aunque más compleja)

Seriabilidad de los planes

PLANES EQUIVALENTES POR CONFLICTOS

Dos **planes** son **equivalentes por conflictos** si el orden de cualesquiera dos operaciones en conflicto es el mismo en ambos planes

PLAN SERIABLE POR CONFLICTOS

Un **plan P** es **seriable por conflictos** si es equivalente (por conflictos) a algún plan en serie P'

- Podremos reordenar las operaciones de P que no estén en conflicto, hasta obtener el plan en serie equivalente P'

» El plan **D** de la figura 4(c) es equivalente al plan A de la fig. 4(a): D es un plan seriable
» El plan **C** de la figura 4(c) no es equivalente a A ni a B, por lo que el plan C NO es seriable

Prueba de la Seriabilidad por Conflictos de un plan

ALGORITMO para comprobar si un plan P es seriable por conflictos

- Considera operaciones LEER_ELEMENTO y ESCRIBIR_ELEMENTO
- Construcción del GRAFO DE PRECEDENCIA dirigido $G=(N, A)$
  - $N=\{T1, T2, \ldots Tn\}$ conjunto de nodos (uno por cada transacción Ti en P)
  - $A=\{a1, a2, \ldots am\}$ conjunto de aristas dirigidas
    - $ai = (Tj \rightarrow Tk)$, con $1 \bullet j \bullet n$, $1 \bullet k \bullet n$
    - si una operación de Tj aparece en el plan antes que alguna operación en conflicto de Tk

Si **hay un ciclo en el grafo, P no es seriable (por conflictos)**

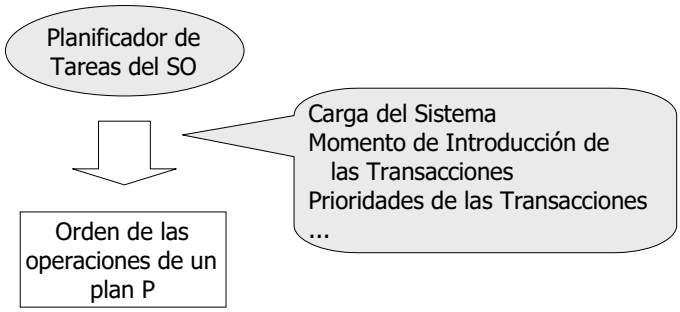
- Un **ciclo** es una secuencia de aristas $C=((Tj \rightarrow Tk), (Tk \rightarrow Tp), \ldots (Ti \rightarrow Tj))$

- $(Tj) \rightarrow (Tk)$ indica que Tj debe ir antes que Tk en un plan en serie equivalente a P (pues dos operaciones en conflicto aparecen en dicho orden en P)
- Si no hay ciclos, es posible crear un plan en serie S equivalente a P, mediante una ordenación topológica de las transacciones

Figuras 6 y 7

Aplicaciones de la Seriabilidad

- Recordando que...
  S en serie $\Rightarrow$ S es correcto,
  P no en serie y equivalente a S $\Rightarrow$ P es seriable
- P es seriable $\Rightarrow$ P es correcto, pero "seriable" ≠ "en serie"
  - un plan seriable permite la concurrencia
- Es muy difícil comprobar por anticipado la seriabilidad de un plan



Aplicaciones de la Seriabilidad

Problema 1:

- Es necesario ENCONTRAR MÉTODOS que GARANTICEN la seriabilidad SIN VERIFICAR la seriabilidad de un plan DESPUÉS de EJECUTARLO
- Método (basado en la TEORÍA DE LA SERIABILIDAD) que define un conjunto de reglas (protocolo) tal que,
  - si TODAS las transacciones las cumplen, o
  - el SGBD las impone

  garantiza la SERIABILIDAD de TODOS los planes en los que participen las transacciones

Aplicaciones de la Seriabilidad

Problema 2:

- En un sistema de procesamiento de transacciones, continuamente se están introduciendo transacciones en el sistema
- Es difícil encontrar planes completos...

¿Cuándo comienza y termina un plan?

La teoría de la seriabilidad considera la
PROYECCIÓN CONFIRMADA de un plan P

La **proyección confirmada C(P) de un plan P** incluye sólo las operaciones de P que pertenecen a transacciones confirmadas

» Un plan P es seriable si su proyección confirmada $C(P)$ es equivalente a algún plan en serie, ya que el SGBD sólo garantiza las transacciones confirmadas

TÉCNICAS de CONTROL de CONCURRENCIA

- Garantizan el aislamiento (no interferencia) de transacciones concurrentes
- Aseguran *planes seriables* empleando protocolos que garantizan la seriabilidad

Lo que vamos a estudiar…

- Tipos de protocolos
  - **Basados en bloqueos**
  - Basados en marcas de tiempo
  - Técnicas de multiversión
  - Protocolos optimistas
- Granularidad de los elementos de información

TÉCNICAS de BLOQUEO: CANDADOS

CANDADO

- Variable asociada a un elemento de información X
- Describe su estado respecto a las operaciones aplicables a X
- Sincroniza el acceso al elemento X por parte de transacciones concurrentes
- El Gestor de Bloqueos (subsistema del SGBD) gestiona y controla el acceso a los candados

TIPOS DE CANDADOS

CANDADOS BINARIOS

CANDADOS DE MODO MÚLTIPLE

TÉC. de BLOQUEO: Candados Binarios

BLOQUEO BINARIO

- Dos *estados* o valores posibles para el candado:
  - **Bloqueado** (1) y
  - **Desbloqueado** (0)

» Si candado(X)=1, ninguna operación de BD que solicite X tendrá acceso a X
» Si candado(X)=0, se podrá acceder a X cuando se solicite

- Las transacciones deben incluir las *operaciones*... (fig. 8)
  - BLOQUEAR(X) para solicitar acceso al elemento X (LOCK)
  - DESBLOQUEAR(X) después de usar el elemento X (UNLOCK)

» EXCLUSIÓN MÚTUA sobre X

TÉC. de BLOQUEO: Candados Binarios

REGLAS EN EL BLOQUEO BINARIO

1. T debe emitir BLOQUEAR(X) antes de que se realice cualquier operación LEER_ELEMENTO(X) o ESCRIBIR_ELEMENTO(X) en T

2. T debe emitir DESBLOQUEAR(X) después de haber completado todas las operaciones LEER_ELEMENTO(X) y ESCRIBIR_ELEMENTO(X) en T

3. T no emitirá BLOQUEAR(X) si ya bloqueó X (posee su candado)

4. T no emitirá DESBLOQUEAR(X) salvo si posee el bloqueo de X

- Sólo UNA transacción puede poseer el candado de X

» Dos transacciones NO pueden tener acceso concurrente al mismo elemento
» BASTANTE RESTRICTIVO!

TÉC. de BLOQUEO: Candados Múltiples

CANDADOS COMPARTIDOS y EXCLUSIVOS

- Tres *estados* posibles para un candado:
  - **Bloqueado para lectura**
  - **Bloqueado para escritura**
  - **Desbloqueado**
- Tres *operaciones* (que las transacciones deben incluir):
  - BLOQUEAR_LECTURA(X)
  - BLOQUEAR_ESCRITURA(X)
  - DESBLOQUEAR(X)
- Un elemento X bloqueado para...
  - lectura → tiene un *CANDADO COMPARTIDO*: otras T's pueden leer X
  - escritura →tiene un *CANDADO EXCLUSIVO*: T posee X en exclusiva

figura 9

TÉC. de BLOQUEO: Candados Múltiples

REGLAS EN EL BLOQUEO DE MODO MÚLTIPLE

1. T debe emitir BLOQUEAR_LECTURA(X) o BLOQUEAR_ESCRITURA(X) antes de que se realice cualquier operación LEER_ELEMENTO(X) en T

2. T debe emitir BLOQUEAR_ESCRITURA(X) antes de realizar cualquier operación ESCRIBIR_ELEMENTO(X) en T

3. T debe emitir DESBLOQUEAR(X) una vez completadas todas las operaciones LEER_ELEMENTO(X) y ESCRIBIR_ELEMENTO(X) de T

4. T no emitirá BLOQUEAR_LECTURA(X) si ya posee un candado de lectura (compartido) o de escritura (exclusivo) para X (*esta regla puede permitir excepciones*)

5. T no emitirá BLOQUEAR_ESCRITURA(X) si ya posee un candado de lectura (compartido) o de escritura (exclusivo) para el elemento X (*esta regla puede permitir excepciones*)

6. T no emitirá DESBLOQUEAR(X) salvo si posee un candado de lectura (compartido) o de escritura (exclusivo) para X

TÉC. de BLOQUEO: Candados Múltiples

EXCEPCIONES en reglas **4** y **5:**

PROMOCIÓN y DEGRADACIÓN de candados

- T emitió BLOQUEAR_LECTURA(X) y luego PROMUEVE el candado emitiendo BLOQUEAR_ESCRITURA(X)
  - OK si T es la **única** que tiene X bloqueado para lectura
- T emitió BLOQUEAR_ESCRITURA(X) y luego DEGRADA el candado emitiendo BLOQUEAR_LECTURA(X)
  - permite que otras transacciones lean X

♣ Emplear candados NO GARANTIZA LA SERIABILIDAD de los planes (fig. 10)

♣ Es necesario seguir un PROTOCOLO adicional que indique DÓNDE COLOCAR las operaciones de BLOQUEO y DESBLOQUEO de candados dentro de las transacciones

BLOQUEO DE DOS FASES (básico)

- T sigue el **protocolo de bloqueo de dos fases** si TODAS las operaciones de BLOQUEO preceden a la PRIMERA operación de desbloqueo

» De este modo, podemos ver T dividida en dos fases:
  - FASE DE EXPANSIÓN (o crecimiento)
    - Se pueden ADQUIRIR candados (o PROMOVER)
    - No se puede LIBERAR ningún candado
  - FASE DE CONTRACCIÓN
    - Se puede LIBERAR candados existentes
    - No se puede ADQUIRIR ningún candado

- Si se permite degradar candados, ha de variarse esta definición:
  - Toda degradación debe realizarse en la fase de contracción
  - » Un BLOQUEAR_LECTURA(X) que degrada un candado de escritura sobre X, s ólo puede aparecer en la fase de contracci ón de T

  figs. 10 y 11

BLOQUEO DE DOS FASES (básico)

Si toda transacción de un plan P sigue el protocolo de bloqueo de dos fases (B2F), entonces P es seriable

☺* Ya no es necesario comprobar la seriabilidad de los planes
- Al imponer las reglas de bloqueo, también se impone la seriabilidad

☹* El B2F puede limitar el grado de concurrencia en un plan
- Es el precio que hay que pagar por garantizar la SERIABILIDAD de TODOS los planes SIN TENER QUE EXAMINARLOS

* Emplear candados puede provocar problemas de ...
- BLOQUEO MORTAL (fig. 12)
- ESPERA INDEFINIDA

BLOQUEO DE DOS FASES

BLOQUEO DE DOS FASES CONSERVADOR (B2F ESTÁTICO)

- T debe bloquear todos los elementos a los que tendrá acceso ANTES de comenzar a ejecutarse
  (PREDECLARACIÓN de su CONJUNTO DE LECTURA y DE ESCRITURA)
  - Si no es posible bloquear alguno, T no bloqueará ningún elemento y esperará
  - Protocolo LIBRE de BLOQUEO MORTAL

BLOQUEO DE DOS FASES ESTRICTO

- T no libera ningún candado antes de terminar (con COMMIT o ROLLBACK)
  - Ninguna otra transacción lee o escribe un elemento escrito por T, salvo si T se ha confirmado ($\rightarrow$ PLAN ESTRICTO)
  - Protocolo NO LIBRE DE BLOQUEO MORTAL (salvo si se combina con B2F conservador)
- Es el protocolo más utilizado

BLOQUEO MORTAL

Dos transacciones esperan que la otra libere su candado sobre cierto elemento de información figura 12

Enfoques para resolver el problema:

PROTOCOLOS DE PREVENCIÓN DEL BLOQUEO MORTAL

- B2F CONSERVADOR
- ORDENAMIENTO
- USO DE MARCAS DE TIEMPO DE TRANSACCIÓN
- PROTOCOLO DE NO ESPERAR
- PROTOCOLO DE ESPERA CAUTELOSA
- USO DE TIEMPOS PREDEFINIDOS

PROTOCOLOS DE DETECCIÓN DEL BLOQUEO MORTAL

- USO DE GRAFOS DE ESPERA

PREVENCIÓN DEL BLOQUEO MORTAL

- **PROTOCOLO DE B2F CONSERVADOR**

  Toda T debe bloquear por adelantado todos los elementos que necesita
  Si no puede bloquearlos todos, espera y lo reintenta después

  » Concurrencia muy limitada

- **ORDENAMIENTO**

  Ordenar los elementos de BD y asegurar que si T necesita varios elementos, los bloqueará en ese orden

  » Concurrencia muy limitada

  » El programador debe conocer el orden elegido --- poco práctico!

PREVENCIÓN DEL BLOQUEO MORTAL

- USO DE **MARCA DE TIEMPO DE TRANSACCIÓN**
  - Si T está implicada en posible bloqueo mortal ¿debe esperar, abortar o hacer que otra T' aborte?
  - Uso de Marca de Tiempo de Transacción MT(T)
    - Identificador único para T
    - Las MT se ordenan según se inician las T
    - La T más antigua tiene la MT(T) menor

Dos esquemas que evitan el bloqueo mortal

- Ti intenta bloquear X, pero X está bloqueado por Tj con un candado en conflicto

1. ESPERAR - MORIR
2. HERIR - ESPERAR

PREVENCIÓN DEL BLOQUEO MORTAL

- **ESPERAR - MORIR**:

  si MT(Ti) < MT(Tj) entonces Ti puede esperar
  si no, se aborta Ti (Ti muere) y
  se reinicia después con la misma marca de tiempo

» Una Ti más antigua espera a que termine otra Tj más reciente
» Una Ti más reciente que solicita un elemento bloqueado por una Tj más antigua, es abortada y reiniciada

- **HERIR - ESPERAR**:

  si MT(Ti) < MT(Tj) entonces se aborta Tj (Ti hiere a Tj) y
  se reinicia después con la misma MT
  si no, Ti puede esperar

» Una Ti más reciente espera a que termine una Tj más antigua
» Una Ti más antigua que solicita un elemento bloqueado por una Tj más reciente, hace que la más reciente sea abortada y reiniciada

PREVENCIÓN DEL BLOQUEO MORTAL

Ventaja

- Protocolos libres de bloqueo mortal
  (ambos hacen que se aborte la T más reciente)

Desventajas

- Ambos hacen que se aborten/reinicien transacciones que PODRÍAN provocar un bloqueo mortal (¡¡aunque tal cosa nunca ocurriera!!)
- (esperar-morir) Una Tj podría abortar y reiniciarse varias veces seguidas si Ti más antigua sigue bloqueando el X que Tj solicita

PREVENCIÓN DEL BLOQUEO MORTAL

- PROTOCOLO DE **NO ESPERAR**

  Si T no puede obtener un candado, es abortada y
  es reiniciada después

  – Al reiniciar T, NO se comprueba si ocurrirá o no un bloqueo mortal

  » Demasiados abortos/ reinicios innecesarios

- PROTOCOLO DE **ESPERA CAUTELOSA**

  Si Tj no está detenida (esperando algún otro elemento bloqueado) entonces
  Ti se detiene y espera
  Si no, se aborta Ti

  – Protocolo libre de bloqueo mortal

- USO DE **TIEMPOS PREDEFINIDOS**

  Si T espera un tiempo > tiempo predefinido por el sistema entonces
  el sistema supone que T está en un bloqueo mortal y aborta T

  – Al abortar T, NO se comprueba si realmente está o no en un bloqueo mortal

DETECCIÓN DEL BLOQUEO MORTAL

- Verificación periódica del estado del sistema

  ¿Está en un estado de bloqueo mortal?

- Conviene **DETECTAR el bloqueo mortal** cuando...

  se sabe que hay poca interferencia entre transacciones

  (*casi nunca* acceden a los mismos datos al mismo tiempo), es decir si...

  - las transacciones son cortas y bloquean pocos elementos, o
  - la carga de transacciones es pequeña

- Pero conviene **PREVENIR el bloqueo mortal** cuando...
  - Las transacciones son largas y bloquean muchos elementos, o si
  - la carga de transacciones es elevada

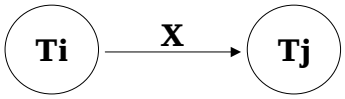
- Uso de GRAFOS de ESPERA para detectar un estado de bloqueo mortal

DETECCIÓN DEL BLOQUEO MORTAL

GRAFOS de ESPERA

- Un NODO por cada transacción en ejecución, etiquetado con el identificador T

- Si Ti espera el elemento X, bloqueado por Tj, crear un ARCO DIRIGIDO desde Ti a Tj



- Cuando Tj libera el candado sobre X, se borra la arista correspondiente

- Si existe algún ciclo en el GRAFO de ESPERA, entonces se ha detectado un estado de bloqueo mortal entre transacciones

DETECCIÓN DEL BLOQUEO MORTAL

- Problema:

  ¿Cuándo verificar el estado del sistema?

  – Cuando haya cierto número de transacciones en ejecución,
  – Cuando varias transacciones en ejecución están cierto tiempo esperando bloquear elementos

  fig. 12

- Si el sistema está en un estado de bloqueo mortal, es necesario abortar algunas transacciones...

  ¿cuáles?

DETECCIÓN DEL BLOQUEO MORTAL

SELECCIÓN DE VÍCTIMAS

- No elegir transacciones que...
  – lleven mucho tiempo en ejecución y
  – hayan realizado muchas modificaciones
- Elegir transacciones que...
  – no hayan realizado muchos cambios
  – participen en varios ciclos de bloqueo mortal en el grafo de espera

- Problema: REINICIO CÍCLICO

  T es abortada, se reinicia y cae en otro bloqueo mortal

  – Solución: asignación de prioridades más altas a las T abortadas varias veces, para no ser siempre las víctimas

ESPERA INDEFINIDA

Una transacción no puede seguir durante un tiempo indeterminado, mientras otras transacciones continúan ejecutándose con normalidad

- Esto ocurre si el esquema de espera da más prioridad a unas transacciones que a otras (ESQUEMA DE ESPERA INJUSTO)

PROTOCOLOS DE **PREVENCIÓN DE ESPERA INDEFINIDA**

- ESQUEMA DE ESPERA JUSTO

- **PRIMERO QUE LLEGA, PRIMERO QUE ES ATENDIDO**
  T puede bloquear el elemento X en el orden en que solicitó su bloqueo
- **AUMENTO DE PRIORIDAD EN LA ESPERA**
  Cuanto más espera T mayor es su prioridad
  Cuando T tiene la prioridad más alta de todas, continúa su ejecución

INANICIÓN

- Problema debido a los algoritmos de resolución del bloqueo mortal

Una transacción es seleccionada para ser abortada (víctima) sucesivamente: nunca termina su ejecución

- Los esquemas ESPERAR-MORIR y HERIR-ESPERAR evitan la inanición

GRANULARIDAD DE LOS DATOS

- Toda técnica de control de concurrencia supone que

  BD = {elementos}

- Un **elemento** puede ser...
  - un REGISTRO de la BD
  - un VALOR de CAMPO de un registro de la BD
  - un BLOQUE de disco
  - un FICHERO
  - la BD completa
- GRANULARIDAD = TAMAÑO DEL ELEMENTO DE INFORMACIÓN
  - GRANULARIDAD FINA
  - GRANULARIDAD GRUESA
- Para cada elemento se tiene un candado distinto

GRANULARIDAD DE LOS DATOS

Ventajas/Desventajas al escoger la granularidad, en el contexto de bloqueo

- ↓ TAMAÑO(elemento) ⇒ ↑ Grado de CONCURRENCIA
- ↓ TAMAÑO(elemento)
  - ⇒ ↑ número de ELEMENTOS en la bd (y de CANDADOS)
  - ⇒ ↑ TRABAJO para el GESTOR DE CANDADOS, y
  - ⇒ ↑ ESPACIO ocupado por la tabla de bloqueo

Pero… ¿cuál es el tamaño adecuado para los elementos?

... pues depende del tipo de las transacciones implicadas ...

- Si T (representativa) accede a POCOS registros
  - elegir granularidad de registro
- Si T accede a MUCHOS registros de un MISMO fichero
  - elegir granularidad de bloque o de fichero