Técnicas de control de concurrencia

Bases de datos II: Tema 4

Índice

- Ordenamiento por marcas de tiempo.
  - Ordenamiento total / Ordenamiento parcial
  - Regla de Thomas
- Técnicas multiversión
- Asignación dinámica de marcas de tiempo.
- Protocolos optimistas de validación.
- Técnicas del bloqueo.
  - Bloqueo en dos fases
  - Interbloqueo
  - Granularidad

Técnicas de control de concurrencia

- Los SGBD incluyen protocolos (conjuntos de reglas) para asegurar la propiedad de aislamiento de transacciones que se ejecutan concurrentemente.
- La mayoría de las técnicas de control de concurrencia aseguran que los planes sean serializables.
- Estas técnicas se aplican conjuntamente a las técnicas de recuperación (Tema 5)

1. Algoritmos basados en marcas de tiempo

- Un enfoque para garantizar la seriabilidad de los planes supone usar marcas de tiempo para ordenar la ejecución de estas.
  - Una marca de tiempo es un identificador único que el SGBD crea para identificar una transacción.
- Las marcas de tiempo se asignan en el orden en que las transacciones se introducen en el sistema.
- Generación de marcas de tiempo:
  - Contador
  - Reloj del sistema
- La marca de tiempo de gránulo es un valor numérico asociado con un gránulo que almacena la marca de tiempo de la última transacción que operó sobre el gránulo.

1. Algoritmo de ordenación total de marcas de tiempo

- Se basa en asegurar que el acceso a los gránulos por las transacciones se realiza en el orden asignado inicialmente (que es el orden de inicio de las transacciones).
- Si esto no se cumple:
  - se debe abortar una transacción (la que produjo el conflicto)
  - se revierte la transacción y se relanza asignándole otra marca tiempo.
- Posible codificación:

```
void opera(transaccion t, granulo g, valor &v)
{ if(marca_tiempo_granulo(g)<=marca_tiempo_transaccion(t))
  {
    realiza_operacion(g,v);
    marca_tiempo_granulo(g)=marca_tiempo_transaccion(t);
  }
  else
       aborta_transaccion();
}
```


1. Algoritmo de ordenación parcial

- Se intenta ordenar aquellas operaciones que son conflictivas (y que no son permutables).
- Se definen dos marcas de tiempo para un gránulo:
  - Marca de tiempo de lectura (MT_lec(g)):
    corresponde a la mayor (la más alta) marca de tiempo de las transacciones que han leído el granulo.
  - Marca de tiempo de escritura (MT_esc(g)):
    corresponde a la mayor (la más alta) marca de tiempo de las transacciones que han escrito en el granulo.
- Este algoritmo comprueba si las operaciones en conflicto respetan el orden asignado a las transacciones.
- Se realizan comprobaciones diferentes para las operaciones de lectura y escritura.

1. Algoritmo de ordenación parcial

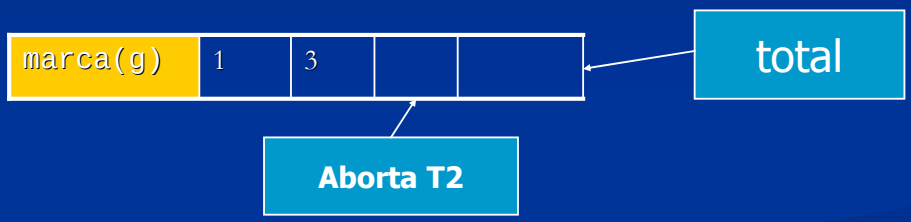
```
void lee(transaccion t,granulo g, valor &v)
{
    if ( marca_g_esc(g)<=marca_transac(t) )
    {
        v = realiza_lectura(g);
        marca_g_lee(g) = max(marca_g_lee(g),marca_transac(t))
    }
    else
        aborta_transaccion(t);
}
void escribe(transaccion t, granulo g, valor v)
{
   if ( marca_g_esc(g)<=marca_transac(t) &&marca_g_lee(g)<=marca_transac(t))
  {
      realiza_escritura(g,v);
      marca_g_esc(g)=marca_transac(t));
  }
   else
      aborta_transaccion(t);
}
```


1. Algoritmo de ordenación parcial

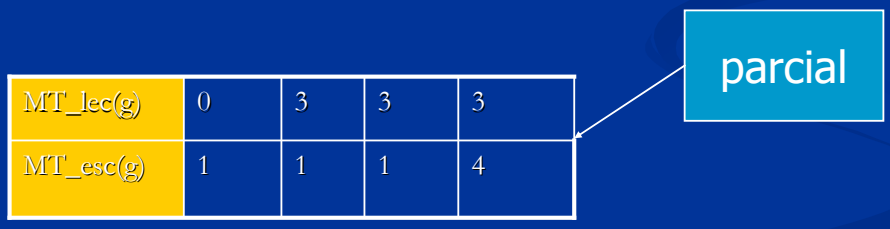
- Este algoritmo detecta operaciones en conflicto que ocurren en orden incorrecto y aborta la T más reciente (mayor marca de tiempo).
- Todas las planificaciones que este algoritmo permite ejecutar son serializables (por conflictos).
- Sin embargo existen planificaciones serializables que no se pueden ejecutar con este algoritmo.
- Es posible que se produzca un reinicio cíclico de alguna de las transacciones (espera indefinida).

1. Ejemplos de algoritmos de ordenación (parcial y total).

T1:esc g, T3: lee g, T2: lee g, T4: esc g



Aborta T2



Regla de Thomas

Regla de escritura de Thomas: constituye una mejora del algoritmo anterior en la escritura. Si una transacción pretende escribir en un gránulo con una MT_esc > MT de la transacción, se puede **ignorar** dicha escritura (siempre que ninguna transacción con marca posterior haya leído el granulo).

```
escribe(t,g,v)
{
  if ( marca_g_esc(g)<=marca_transac(t) &&
   marca_g_lee(g)<=marca_transac(t) )
  {
    lleva_a_cabo_escritura(g,v);
    marca_g_esc(g)=marca_transac(t));
  }else if ( marca_g_lee(g)<=marca_transac(t) &&
          marca_g_esc(g)>marca_transac(t) )
           ignora_escritura();
        else
           aborta_transaccion();
}
```

1. Ejemplos de algoritmos de ordenación (parcial con regla de Thomas).

```
T1:esc g, T1: lee g, t3: esc g, t2: esc g, t4: lee g
```

MT_lec(g)	0	1	1	
MT_esc(g)	1	1	3	

Parcial
(sin Regla de Thomas)

Aborta T2

MT_lec(g)	0	1	1	Se ignora escritura T2	4
MT_esc(g)	1	1	3		3

Parcial
(con Regla de Thomas

Algoritmo multiversión

- Este algoritmo consiste en ir guardando varias versiones del mismo dato (gránulo) : se conservan los valores antiguos de los gránulos que se han actualizado.
- Lectura:

  Cuando una Transacción necesita leer algún gránulo, se elige una versión adecuada para mantener la seriabilidad de la planificación, si es posible.
- Escritura:

  Cuando una Transacción escribe un gránulo, escribe una nueva versión de ese gránulo, conservándose además la versión anterior.
- Desventaja: requiere más almacenamiento que los anteriores.
  - Muchas veces estas versiones se aprovechan para la recuperación.
- Abortar transacciones: se aborta la transacción que provocó el conflicto (la más reciente) y se le asigna una nueva marca de tiempo.
- *Este algoritmo se basa en el concepto de la seriabilidad por vistas.*

Algoritmo multiversión: codificación

```
void lee(transaccion t,granulo g,valor &v)
{
   marca k;

   //encuentra la version más alta <= marca_transac(t)
   k=ultima_version(g);
   while(marca_g_esc(g,k) > marca_transac(t))
   {
        k=version_previa(g,k) )
   }

   v=lleva_a_cabo_lectura(g,k);
   marca_g_lee(g,k) = max(marca_g_lee(g,k),marca_transac(t));
}
```

Las lecturas siempre tienen éxito

Algoritmo multiversión: codificación

```
void escribe(transaccion t,granulo g,valor v)
{
  marca k;

  k=ultima_version(g);
  while(marca_g_esc(g,k)>marca_transac(t))
       k=version_previa(g,k);

  if ( marca_g_lee(g,k) <= marca_transac(t) )
   {
       lleva_a_cabo_escritura(g,v);
       inserta_valor(k,v);
       inserta_marca_esc(k);
   }
   else
       aborta_transaccion();
}
```

Las escrituras pueden abortar T

Algoritmo multiversión: ejemplo

t1:esc g, t4: lee g, t5: lee g, t2: lee g

No aborta transacciones

m_lec	0
m_esc	1
valor	v1

t1:esc g

m_lec	4
m_esc	1
valor	v1

t4: lee g

m_lec	5
m_esc	1
valor	v1

t5: lee g

m_lec	5
m_esc	1
valor	v1

t2: lee g

t1:esc g, t4: lee g, t5: lee g, t2: esc g

Aborta T2!!!

m_lec	0
m_esc	1
valor	v1

t1:esc g

m_lec	4
m_esc	1
valor	v1

t4: lee g

m_lec	5
m_esc	1
valor	v1

t5: lee g

Algoritmo multiversión: ejemplo

t1:esc g, t4: esc g, t5: lee g, t2: lee g

hay 2 versiones

M_lec	0
M_esc	1
valor	v1

M_lec	0	0
M_esc	1	4
valor	v1	v4

M_lec	0	5
M_esc	1	4
valor	v1	v4

M_lec	2	5
M_esc	1	4
valor	v1	v4

T1:esc g

t4: esc g

t5: lee g

t2: lee g

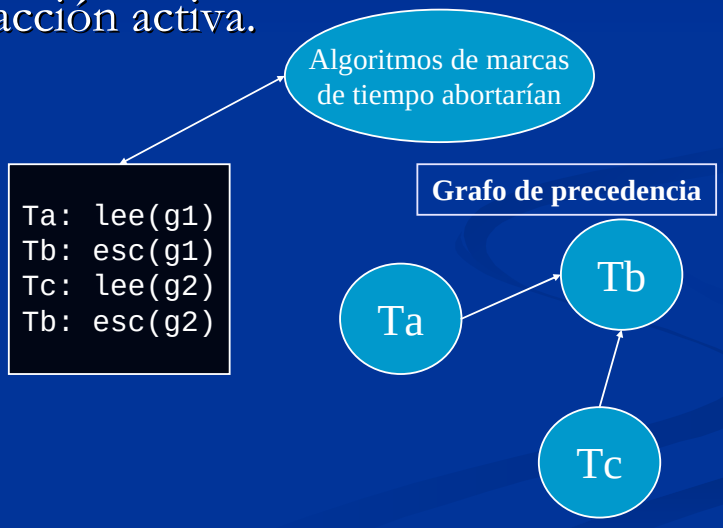
¿qué haría el Algoritmo de OMT parcial

M_lec	0	0	5	
M_esc	1	4	4	

Aborta T2

Asignación dinámica de marcas de tiempo

- Las marcas de tiempo se asignan dinámicamente y no por simple orden de inicio
- Detección de conflictos: se comprueba si el gránulo al que accede una transacción dada ha sido accedido por una transacción activa.
- Ejemplo:



Asignación dinámica

Posible algoritmo:

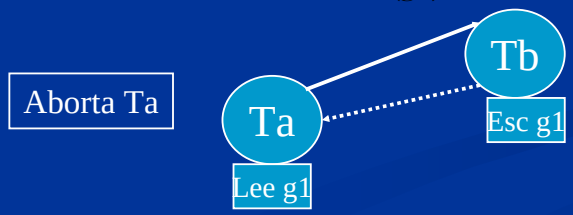
1) Nueva transacción : crea nodo.

2) Operación sobre gránulo g, añade gránulo a la lista de la transacción (con tipo) y establece relación de precedencia con todas las transacciones que ya lo hayan afectado (salvo que ambas sean lecturas). Comprueba si hay ciclos. Si los hay, se aborta la transacción que ha creado el conflicto.

3) Termina transacción marca nodo como terminado, pero sólo se elimina si no llegan flechas a él. Si se elimina, se busca si hay nodos conectados directamente a él que también sean iniciales y se eliminan a su vez.

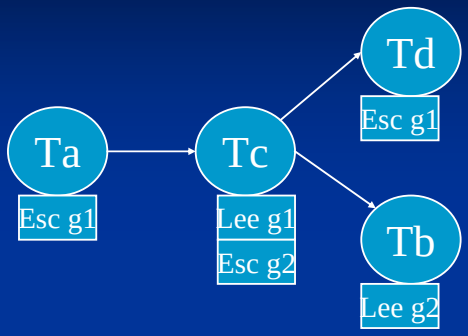
Ejemplo 1:

Ta: init, Tb: init, Ta: lee(g1), Tb: esc(g1), Ta: esc(g1)

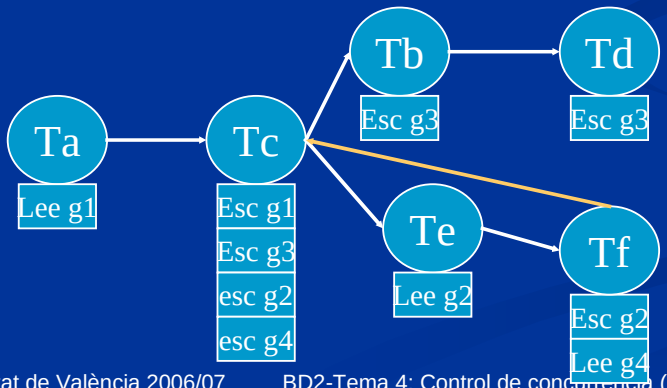


Asignación dínámica: ejemplos

{ T_a init, T_b init, T_a esc g_1, T_c init, T_c lee g_1, T_d init, T_d esc g_1, T_c esc g_2, T_b lee g_2 }



{ T_a init, T_b init, T_c init, T_d init, T_a lee g_1, T_c esc g_1, T_e init, T_c esc g_3, T_c esc g_2, T_e lee g_2, T_b esc g_3, T_d esc g_3, T_f esc g_2, T_f lee g_4, T_c esc g_4 }



Se aborta Tc, pero sería más conveniente abortar Tf

Algoritmos optimistas

- No realizan ninguna verificación durante la ejecución.
- Los cambios se realizan sobre copias locales (no sobre los gránulos de la BD).
- Al final de la ejecución, existe una fase de validación que comprueba si cualquiera de las actualizaciones violaba la seriabilidad.
- Este algoritmo tiene tres fases:
  - Fase de lectura: leo las valores de los gránulos en la BD pero no modifica ningún gránulo (copias locales).
  - Fase de validación: se efectúa una verificación para comprobar si hay algún problema con las operaciones realizadas.
  - Fase de escritura: si la fase anterior termina con éxito se actualiza la BD.

Algoritmos optimistas

- Si hay pocas interferencias (conflictos) entre transacciones (mayoritariamente son lecturas), casi todas se validarán sin dificultad (por ello se llaman optimistas).
- Existen varios algoritmos de este tipo. Uno de ellos se basa en la asignación de marcas de tiempo.
- La fase de validación comprueba para cada transacción Ti las siguientes condiciones:

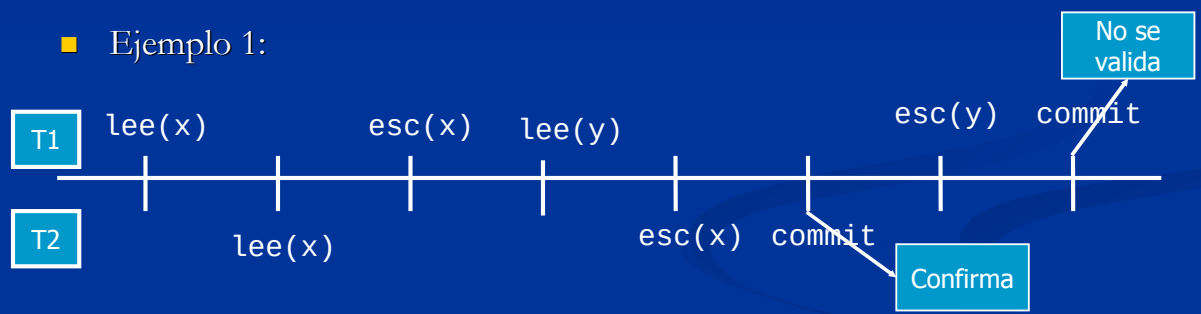
1. **La transacción Tj acaba antes de que Ti comience**
2. **Si una transacción Ti comienza antes de que Tj acabe,**
   1. **El conjunto de datos escritos por Tj (transacción previa) no son los datos leídos por la transacción Ti (actual).**
   2. **La transacción Tj (previa) completa la fase de escritura antes de que la transacción Ti valide.**

Algoritmos optimistas

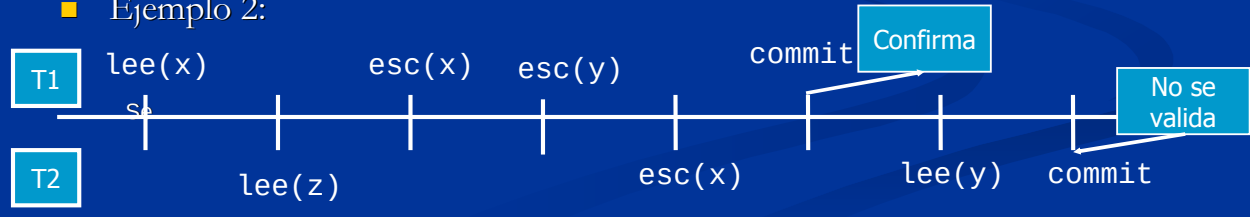
- Las condiciones anteriores se resumen en:

  Se comprueba si existe algún gránulo que haya sido leído por la transacción a certificar y escrito por otra transacción certificada después del comienzo de la transacción que estamos certificando.

- Ejemplo 1:



- Ejemplo 2:



Técnicas de bloqueo

- Las técnicas más empleadas para controlar el acceso concurrente de las transacciones se basan en el concepto de bloquear elementos de datos.
- Un bloqueo corresponde a una variable asociada al gránulo que describe las operaciones que se pueden realizar sobre él.
- Dado que dos operaciones de transacciones diferentes sobre el mismo gránulo que no sean permutables pueden provocar violaciones en el orden de serialización, pueden prevenirse las violaciones bloqueando los accesos a dicho gránulo.

Técnicas de bloqueo

- Matriz binaria de compatibilidad: Es una matriz binaria que describe las operaciones compatibles (1 = sí, 0 = no).

C	leer	escribir
leer	1	0
escribir	0	0

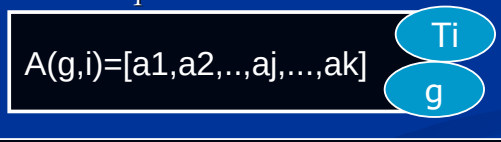
- Se introducen operaciones adicionales en las transacciones:
  - **bloquear(g, M)**: indica al planificador el comienzo de una operación o conjunto de ellas de modo definido por M sobre el gránulo g.
  - **desbloquear(g)**: indica al planificador el final de las operaciones realizadas por la transacción correspondiente.
- Un modo de operación M es un vector binario que indica las operaciones que una T desea bloquear sobre el gránulo g.
  - p.e., si leer o escribir modo M=(1 0) o (0 1) o (1 1)

Técnicas de bloqueo

- Una transacción en el esquema del bloqueo deberá:
  - Emitir una operación bloquear(g,M) antes de operar sobre g en el modo M.
  - Emitir una operación desbloquear(g) después de terminar la operación o operaciones correspondientes.
  - No emitirá una operación bloquear si ya tiene el bloqueo sobre el gránulo (*excepto promociones*).
  - No emitirá una operación desbloquear a menos que ya posea el bloqueo correspondiente.
- El módulo de gestión de bloqueos del SGBD se ocupa de asegurar que se cumpla lo anterior.

Técnicas de bloqueo

- ¿Qué información se necesita mantener en el protocolo del bloqueo?
  - Información sobre las transacciones que bloquean un determinado gránulo: tabla de bloqueo.



aj=1 si Ti tiene bloqueado g para esa operación.
aj=0 si Ti no tiene bloqueado g para esa operación.

  - Las operaciones solicitados para una transacción se almacenan en un vector de bits M (los k valores son las diferentes operaciones posibles, normalmente 2, lee y esc):

M=[m1,m2,..,mj,...,mk]

Técnicas de bloqueo

- Proposición: los modos de operación solicitados durante una acción de bloquear(g, M, Tp) son compatibles con los modos de operación actualmente bloqueados si se cumple lo siguiente:

$$\neg((U_{i!=p} A(i,g)) * \neg C) \supset M$$

- ¿Qué hacer cuando las operaciones no son compatibles? Se guarda la petición en una cola de espera, y se bloquea la transacción hasta que el gránulo este disponible.
  - *La inserción en la cola de espera puede seguir diversos criterios*
- Cada gránulo tiene una cola de espera que indica las transacciones que están esperando para utilizarlo y el modo de operación que solicitan.

Técnicas de bloqueo: codificación

```
bloquear(gránulo g, modos M, id_transac p)
{
if ( ¬((U_{i!=p} A(i,g))*¬C)) ⊃ M )
       A(g,p)=A(g,p) U M;  //permite promociones de bloqueos
   else
   {
       inserta_modo_en_cola(p,M,g);
       bloquea_transaccion(p);
   }
}
```

Técnicas de bloqueo: codificación

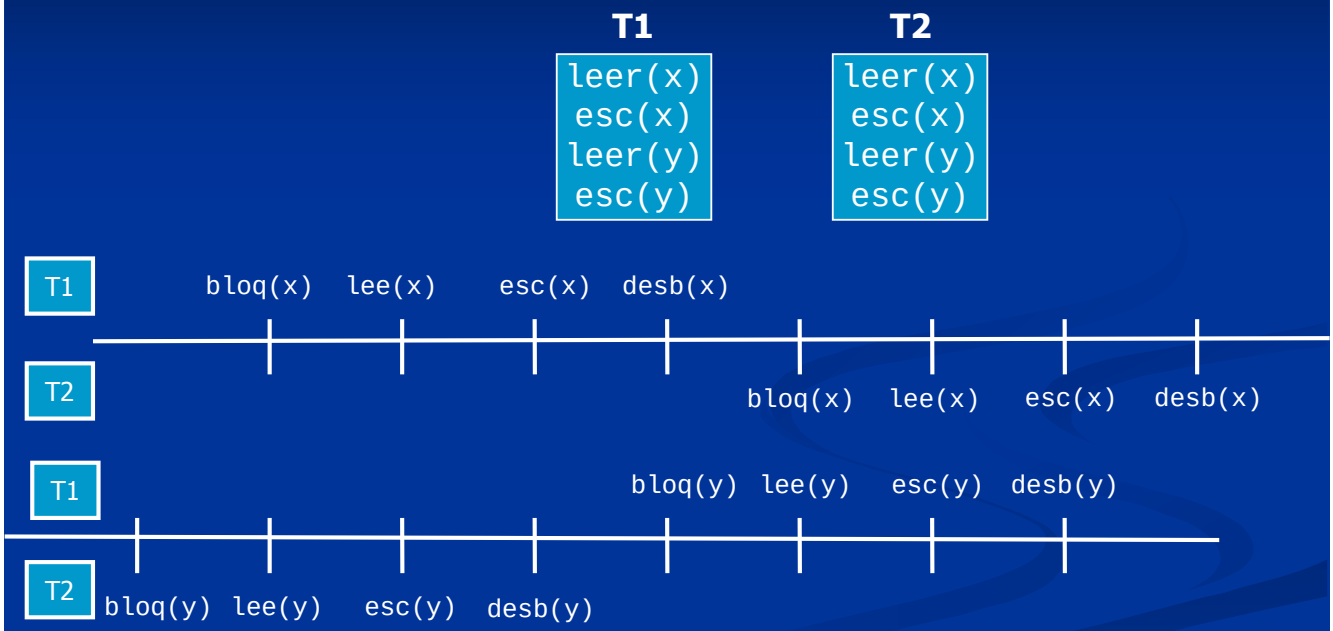
```
desbloquear(gránulo g, id_transac t)
{
  elem_cola *q;
  id_transac p;

  A(g,t)=0; //vector con todo ceros
  q=primer_elem_cola(g);
  while(q!=NULL)
  {
     p=q->id_transac;
     M=q->modo_op;
     if ( ¬((U i!=p A(i,g))*¬C)) ⊃ M )
     {
        A(g,p)=A(g,p) U M;
        eliminar_de_cola(g,p);
        desbloquear_transaccion(p);
     }
     q=sig_elem_cola(g);
   }
}
```


bloqueos y seriabilidad

- Existen ciertas restricciones en el uso del bloqueo y del desbloqueo.
- Se dice que **una transacción está bien formada** si bloqueo todos los modos de operación adecuados antes de operar sobre ellos y los desbloquea después.
- El uso de bloqueos por sí solo no asegura que una planificación sea serializable.

bloqueos y seriabilidad

- Ejemplo: planificación con bloqueos no serializable.



bloqueos y seriabilidad

T1	T2
Bloquea(y,lec)	Bloquea(x,lec)
Lee(y)	Lee(x)
Desbloquea(y)	Desbloquea(x)
Bloquea(x,esc+lec)	Bloquea(y,esc+lec)
Lee(x)	Lee(y)
x=x+y	y=x+y
Escribe(x)	Escribe(y)
Desbloquea(x)	Desbloquea(y)

Ej: x=20,y=30 {T1,T2} x=50,y=80

Ej: x=20,y=30 {T2,T1} x=70,y=50

Ej: x=20,y=30 $P_{conc\_derecha}$ x=50,y=50
es una planificación no serializable

T1	T2
Bloquea(y,lec)	
Lee(y)	
Desbloquea(y)	
	Bloquea(x,lec)
	Lee(x)
	Desbloquea(x)
	Bloquea(y,esc+lec)
	Lee(y)
	y=x+y
	Escribe(y)
	Desbloquea(y)
Bloquea(x,esc+lec)	
Lee(x)	
x=x+y	
Escribe(x)	
Desbloquea(x)	

Bloqueo en dos fases

- Se llama **transacción en dos fases** aquella que no realiza ningún bloqueo después de haber realizado alguna operación de desbloquear. En una transacción en dos fases
  - **fase de expansión** (o de crecimiento), durante la cual se pueden adquirir nuevos bloqueos sobre elementos pero no se puede liberar ninguno.
  - **fase de contracción**, durante la cual se pueden liberar todos los bloqueos existentes pero no se pueden adquirir nuevos bloqueos.
- Proposición:

  toda planificación compuesta de transacciones en dos fases es serializable.

Bloqueos en 2 fases

T1	T2
Bloquea(y,lec)	Bloquea(x,lec)
Lee(y)	Bloquea(y,esc+lec)
Bloquea(x,esc+lec)	Lee(x)
Lee(x)	Desbloquea(x)
x=x+y	Lee(y)
Escribe(x)	y=x+y
Desbloquea(y)	Escribe(y)
Desbloquea(x)	Desbloquea(y)

Ambas son transacciones en 2 fases, por lo que cualquier planificación que pueda suceder es serializable. Por ejemplo:

T1	T2
Bloquea(y,lec)	
Lee(y)	
Bloquea(x,esc+lec)	
	Bloquea(x,lec)
Lee(x)	*(se pone en espera*
x=x+y	*hasta que T1*
Escribe(x)	*desbloquee x)*
Desbloquea(y)	
Desbloquea(x)	
	Bloquea(y,esc+lec)
	Lee(x)
	Desbloquea(x)
	Lee(y)
	y=x+y
	Escribe(y)
	Desbloquea(y)

Bloqueo en dos fases

- El protocolo en dos fases limita la concurrencia pero garantiza que los planes sean serializables.
  - B2F básico: va tomando bloqueos y luego los va liberando.
  - B2F conservador: toma todos los bloqueos al principio y si no se espera *(poco práctico)*.
  - B2F estricto: no libera ningún bloqueo exclusivo hasta después de confirmar o abortar. (asegura planificaciones estrictas)
  - B2F riguroso: no libera ningún bloqueo hasta después de confirmar o abortar.

Problemas debidos al protocolo en dos fases:

- **Problema del interbloqueo:**

es una situación en la que los gránulos han sido bloqueados en una secuencia tal que un grupo de transacciones que no pueden seguir porque están esperando a las otras.

Caracterización del interbloqueo:

1. todas las transacciones están bloqueadas esperando el acceso a un gránulo.
2. la finalización de cualquier otra transacción no permite desbloquear a ninguna transacción.

Posible solución: cada transacción sólo puede bloquear un gránulo a un tiempo (no compatible con transacciones en dos fases)

Bloqueos en 2 fases: interbloqueo

T1	T2
Bloquea(y,lec)	Bloquea(x,lec)
Lee(y)	Bloquea(y,esc+lec)
Bloquea(x,esc+lec)	Lee(x)
Lee(x)	Desbloquea(x)
x=x+y	Lee(y)
Escribe(x)	y=x+y
Desbloquea(y)	Escribe(y)
Desbloquea(x)	Desbloquea(y)

T1	T2
Bloquea(y,lec)	
Lee(y)	
	Bloquea(x, lec)
Bloquea(x,esc+lec)	
(se pone en espera hasta que T2 desbloquee x)	Bloquea(y,esc+lec) *(se pone en espera hasta que T1 desbloquee x)*

Ambas transacciones son en transacciones bien formadas en dos fases.

Pero ninguna de las dos transacciones pueden continuar

Problemas debidos al protocolo en dos fases:

- **Problema de la espera indefinida:**
- cuando un conjunto de transacciones con modos de operación compatibles entre sí impiden que otra transacción con operación no compatible acceda al gránulo.
  - Ejemplo:

```
S={T1:lee(x),T2:lee(x),T3:esc(x), T4:lee(x),
..., Tn:lee(x)}
```

    - La transacción T3 no se desbloquea mientras haya transacciones que tienen el gránulo bloqueado para lectura

Problemas debidos al protocolo en dos fases

- **Problema de los fantasmas:**
- imaginemos una tabla que contiene información de empleados, con tipo y edad:

idempleado	tipo	edad
1	1	35
2	1	50
3	2	45
4	2	55
5	1	40

T1 consulta la tabla para encontrar el empleado más viejo tipo 1 y tipo 2.

T2 inserta uno nuevo empleado tipo 1 de edad 60, y borra el empleado tipo 2 más viejo.

Problemas debidos al protocolo en dos fases

- **Problema de los fantasmas:**

T1	T2
Bloquea(gr.Tipo1, sel)	Inserta(6, 1, 60)
Selecciona max Tipo1	Bloquea(gr.Tipo2, sel)
Bloquea(gr.Tipo2, sel)	Selecciona max Tipo2
Selecciona max Tipo2	Bloquea(gr.4, del)
Desbloquea(gr.Tipo1)	Borra(4, 2, 55)
Desbloquea(gr.Tipo2)	Desbloquea(gr.Tipo2)
	Desbloquea(gr.4)

{T1,T2} daría MaxTipo1=50, MaxTipo2=55

{T2,T1} daría MaxTipo1=60, MaxTipo2=45

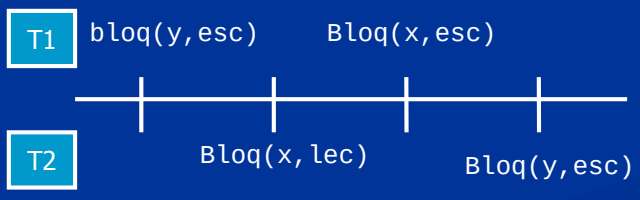
P *(v.derecha)* da MaxTipo1=60, MaxTipo2=55

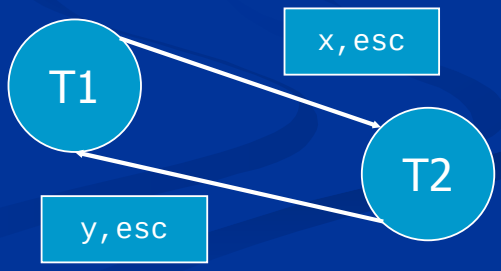
T1	T2
	Inserta(6, 1, 60)
Bloquea(gr.Tipo1, sel)	
Selecciona max Tipo1	
Bloquea(gr.Tipo2, sel)	
Selecciona max Tipo2	
	Bloquea(gr.Tipo2, sel)
	Selecciona max Tipo2
	Bloquea(gr.4, del)
Desbloquea(gr.Tipo1)	
Desbloquea(gr.Tipo2)	
	Borra(4, 2, 55)
	Desbloquea(gr.Tipo2)
	Desbloquea(gr.4)

Soluciones al interbloqueo

- Un esquema para resolver el interbloqueo es su detección.
- Formas de representar el interbloqueo:
  - Grafo de esperas.
  - Grafo de reservas.
- Grafo de esperas:

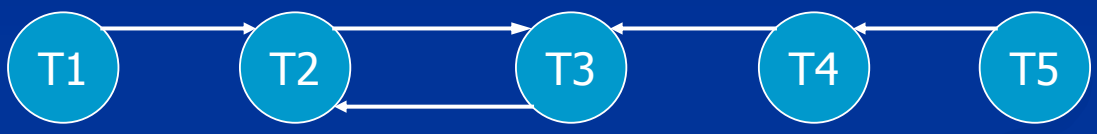
  es un grafo en el que los nodos son las transacciones y la relación de espera entre nodos se define como sigue: una transacción Tp espera a otra transacción Tq si Tp ha solicitado el bloqueo de un gránulo y esta petición no puede ser aceptada porque Tq lo tiene bloqueado.





Soluciones al interbloqueo

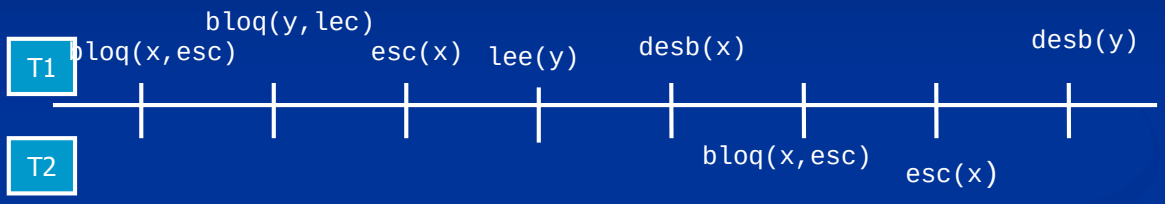
- Ejemplo de grafo de esperas que representa una situación de interbloqueo:



- Proposición: existe una situación de interbloqueo si y solo si hay un ciclo.
- Nota: el hecho de que una transacción espere a otra implica una relación de precedencia.
  Al revés (relación de prec. implica espera) no es cierto.

Soluciones al interbloqueo

- Ejemplo: T1 precede a T2 pero no hay esperas.

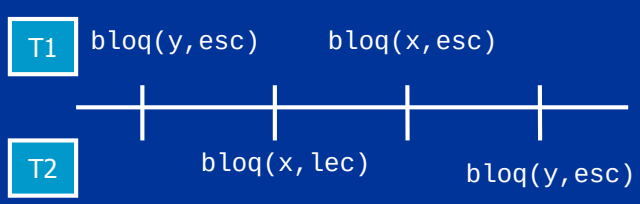


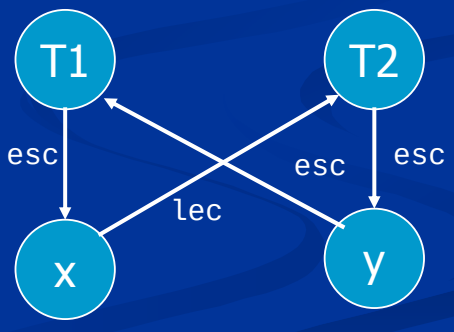
- Si se produce un ciclo en el grafo de esperas, entonces existe un ciclo en el grafo de precedencia y la planificación no es serializable.

Soluciones al interbloqueo

- Grafo de reservas:

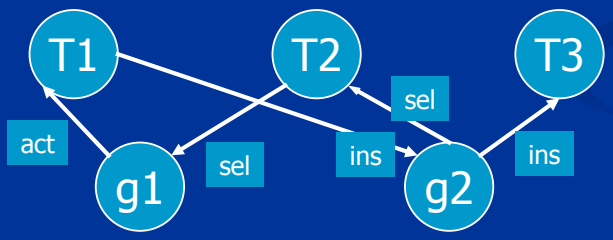
  Grafo con dos tipos de nodos, los de las transacciones y los correspondientes a gránulos.
  - Un arco une un gránulo gi a una transacción Tp si Tp tiene bloqueado el gránulo gi.
  - Un arco une una transacción Tq con un gránulo gj si Tq ha solicitado el bloqueo del gránulo pero no lo ha conseguido.





Soluciones al interbloqueo

- Una condición necesaria para que haya interbloqueo es que exista un ciclo en el grafo de reservas. La condición no es suficiente.
- Ej: ciclo en el grafo de reservas pero sin interbloqueo
  - T1:bloquea(g1,act), T2:bloquea(g2,sel), T2: bloquea(g1,sel), T3:bloquea(g2,ins), T1:bloquea(g2,ins),
    cuando T3:desbloquea(g2) T1 ya puede seguir ,
    cuando T1:desbloquea(g1) T2 ya puede seguir …



C=

act	sel	ins
0	0	0
0	1	1
0	1	0

- Cuando los únicos modos de operación son lectura y escritura la condición es también suficiente.

Detección del interbloqueo

- Existen diversos algoritmos para ello basados en la detección de ciclos en el grafo de esperas, entre ellos:
  - **Algoritmo 1**: Comprueba la existencia de ciclos mediante la eliminación de nodos terminales.

    1. Elimino transacciones de las que no salen flechas.
    2. Elimino flechas que lleguen a ellas.
    3. Repito 1) y 2) hasta que no queden transacciones

  - **Algoritmo 2**: Comprueba posible ciclos desde la última transacción bloqueada y marcando los nodos por los que pasa. Si pasa dos veces por el mismo nodo ha detectado un ciclo.
- Es importante la selección de víctimas.
- Debe determinarse cuándo comprobar si hay interbloqueo

Prevención del interbloqueo

- Las técnicas de interbloqueo utilizan el concepto de marca de tiempo de transacción.
- Existen dos esquemas que evitan el interbloqueo (supongamos Ti quiere acceder a un gránulo bloqueado por Tj)
  - **Esperar-morir**: Si Ti < Tj entonces Ti puede esperar; en caso contrario se aborta Ti y se reinicia con la misma marca de tiempo.
  - **Herir-esperar**: Si Ti < Tj entonces se aborta Tj y se reinicia con la misma marca de tiempo. En caso contrario, Ti puede esperar.

T más antigua puede esperar a una más reciente.

T más reciente puede esperar a una más antigua.

En ambos casos en caso de abortarse se aborta la más reciente

Granularidad de los datos

- Se llama granularidad de los elementos de datos al tamaño de dichos elementos.
- Selección de un tamaño:
  - Mayor tamaño, menor es el grado de concurrencia permitido.
  - Menor tamaño, más elementos de datos, el sistema gestor debe mantener y gestionar más bloqueos.
- ¿Cuál es el mejor tamaño?
  - Depende de los tipos de transacciones implicadas.
- Nivel de granularidad múltiple: una buena opción es que el sistema permita varios niveles de granularidad.
  - Bloqueos en modo de intención

Granularidad de los datos

- Para trabajar con granularidad múltiple se define una matriz de compatibilidad con los modos normales y los modos de intención, repitiéndose C y siendo E una matriz de unos.
  - Por ejemplo:

C	C
C	E

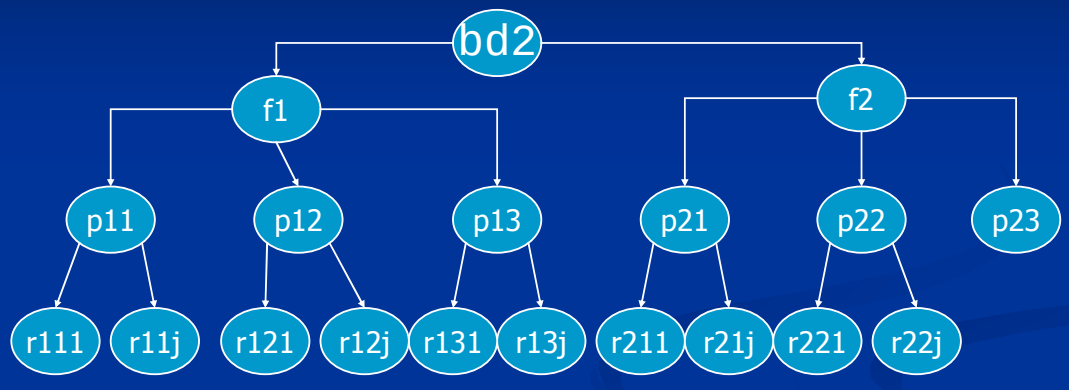
	L	E
L	1	0
E	0	0

	L	E	IL	IE
L	1	0	1	0
E	0	0	0	0
IL	1	0	1	1
IE	0	0	1	1

- Para bloquear un gránulo en un modo determinado, la transacción debe bloquear todos los gránulos que lo contengan en modo de intención y el gránulo en modo normal.

Granularidad de los datos

- Ejemplo que ilustra los bloqueos en modo intención.



T1
`esc(r111)`
`esc(r211)`

T2
`lee(r11j)`
`Lee(p11)`

T3
`esc(p12)`
`lee(f2)`

Granularidad de los datos

- Considérese la siguiente planificación:
  P={ T1:esc(r111), T2:lee(r11j), T3: esc(p12), T1:esc(r211), T2:lee(p11), T3: lee(f2) }
- El funcionamiento sería (A indica el bloque aplicado)
  1. A(f1,T1)=(0001) A(p11,T1)=(0001) A(r111,T1)=(0100)
  2. A(f1,T2)=(0010) A(p11,T2)=(0010) A(r11j,T2)=(1000)
  3. A(f1,T3)=(0001) A(p12,T3)=(0100)
  4. A(f2,T1)=(0001) A(p21,T1)=(0001) A(r211,T1)=(0100)
  5. A(f1,T2)=(0010)
     el bloqueo(p11,T2,1000) es incompatible con A(p11,T1)=(0001) ESPERA
  6. el bloqueo(f2,T3,1000) es incompatible con A(f2,T1)=(0001) ESPERA
  - cuando T1 desbloquee p11 entonces T2 se "despierta" y podrá seguir …
  - cuando T1 desbloquee f2 entonces T3 se "despierta" y podrá seguir …

Bibliografía

- Ramez A. Elmasri, Shamkant B. Navathe, "Fundamentos de Sistemas de Bases de Datos". Addison Wesley (tercera edición, capítulo 20)
- R. Ramakrishnan, J. Gehrke "Database Management Systems". McGraw Hill

Ejercicios propuestos

1. Para cada uno de las siguientes planificaciones, di a qué tipo de planificación pertenece entre las siguientes: serializable (en vistas o en conflictos), recuperables, sin abortos en cascada.

1. `T1:lee(x), T2:lee(x), T1:esc(x), T2:esc(x)`
2. `T1:esc(x), T2:lee(y), T1:lee(y), T2:lee(x)`
3. `T1:lee(x), T2:lee(y), T3:esc(x), T2:lee(x), T1:lee(y)`
4. `T1:lee(x), T1:lee(y), T1:esc(x), T2:lee(y), T3:esc(y), T1:esc(x), T2:lee(y)`
5. `T1:lee(x), T2:esc(x), T2:abort , T1:commit`
6. `T1:lee(x), T2:esc(x), T1:esc(x), T2:commit, T1:commit`
7. `T1:esc(x), T2:lee(x), T1:esc(x), T2:abort , T1:commit`
8. `T1:esc(x), T2:lee(x), T1:esc(x), T2:commit, T1:commit`

2. Indica qué algoritmos de control de concurrencia permitirían la ejecución de las planificaciones del ejercicio anterior (bloqueo con protocolo en dos fases, multiversión, ordenación parcial de marcas de tiempo con y sin reglas de Thomas).

3. Considera las siguientes secuencias de acciones:

P1:{T1: lee(x), T2:esc(x), T2: esc(y), T3: esc(y), T1: esc(y), T1: commit, T2: commit, T3: commit}

P2:{T1: lee(x), T2: esc(y), T2:esc(x), T3: esc(y), T1: esc(y), T1: commit, T2: commit, T3:commit}

Simula el comportamiento de los siguientes algoritmos: optimista, multiversión, protocolo en 2 fases con prevención y detección del interbloqueo.

4. Dar un ejemplo de diferentes planificaciones: serializables en vistas y en conflictos, recuperables, sin cascada.
5. Dado el siguiente esquema

```
emp(id_emp, nombre, edad, salario, id_dept)
dept(id_dept, mobre, piso)
```

sobre el se realiza una actualización en el empleado con nombre 'Ricardo' que incrementa su salario en un 10%

Da un ejemplo de consulta que entraría en conflicto si se ejecutara al mismo tiempo.

6. Indicar el comportamiento del protocolo de asignación dinámica por marcas de tiempo en la siguiente planificación:

P: {T1:init, T1:lee g1, T2:init, T2:lee g2, T2:lee g1, T2:esc g1, T2:fin, T3:init, T3: lee g3, T3:lee g1, T4: init, T4: esc g3, T1: esc g1, ...}

Cuestiones de exámenes anteriores

1. Las planificaciones serializadas no producen inconsistencias en la BD. ¿Existe algún algoritmo de control de concurrencia que aborte transacciones en una planificación serializada?
2. El algoritmo de control de concurrencia por certificación (optimista) aborta todas aquellas transacciones que realizan lectura en gránulos escritos por transacciones que certificaron durante su tiempo de vida. ¿Crees que siempre que se da la situación anterior es necesario abortar la transacción?
3. Las situaciones de interbloqueo pueden evitarse haciendo que cada transacción bloqueo cada gránulo justo antes de usarlo y lo desbloquee justo después de utilizarlo. Sin embargo, el protocolo utilizado por todos los SGBD es el protocolo en dos fases, ¿Por qué?

4. Para conseguir granularidad múltiple en los algoritmos basados en bloqueos, se define lo que se conoce por ‘modos de intención’. Explicar que significa que un determinado gránulo esté bloqueado en modo de intención en lugar de bloqueado en modo normal.
5. Los grafos de reserva se utilizan para la detección del interbloqueo. Una condición necesaria para que exista interbloqueo es que exista un ciclo en el grafo de reservas. ¿Es suficiente esta condición? ¿Por qué?