Concurrencia

M. Andrea Rodríguez-Tastets

Universidad de Concepción,Chile
www.inf.udec.cl\ ∼andrea
andrea@udec.cl

II Semestre - 2014

Objetivos de la unidad

Entender los diferentes protocolos de manejo de concurrencia.

Técnicas de manejo de concurrencia

- Pesimistas: bloqueo y marcas de tiempo (timestamping)
- Optimistas

Técnicas de bloqueo

- Usa una variable asociada a cada elemento de datos que describe el estado de dicho elemento respecto a las posibles operaciones (recuperación o actualización) que se pueden realizar sobre ellos en cada momento.
- Las transacciones llevan a cabo bloqueos, impidiendo a otros usuarios la recuperación o actualización de los elementos bloqueados, para evitar inconsistencias en el acceso concurrente.
- Los SGBD tienen bloqueos (por tupla, por tabla) para asegurar la consistencia. Los usuarios también pueden bloquear explícitamente los objetos, impidiendo el acceso por parte de otros usuarios.

Tipos de bloqueo

- Exclusivos o binario (X): cuando una transacción mantiene un bloqueo de este tipo, ninguna otra transacción puede acceder al objeto bloqueado, ni bloquearlo, hasta que sea liberado por la transacción que lo había retenido. Se utiliza cuando se quiere actualizar datos.
- Bloqueo compartido (S): cuando una transacción bloquea en este modo, permite que otras transacciones retengan también el objeto en bloqueo compartido, pero no exclusivo. Este tipo se utiliza cuando no se requiere actualizar datos, pero se desea impedir cualquier modificación mientras los datos son consultados.

Esquema de bloqueos

- Una transacción T debe emitir un bloqueo de lectura (compartido) sobre X o bloqueo de escritura (exclusivo) sobre X antes de que realice cualquier operación de lectura de X de T.
- Una transacción T debe emitir un bloqueo de escritura (exclusivo) sobre X antes de que realice cualquier operación de escritura de X de T.
- Una transacción T debe emitir la operación desbloquear(X) una vez que se hayan completado las operaciones de leer o escribir X.

Esquema de bloqueos (cont.)

- Una transacción no emitirá una operación de bloqueo de lectura sobre X si ya posee un bloqueo de lectura compartido o de escritura sobre el elemento X. Esto tiene algunas excepciones.
- Una transacción no emitirá una operación de bloqueo de escritura sobre X si ya posee un bloqueo de lectura compartido o de escritura sobre el elemento X.
- Una transacción T no emitirá una operación de desbloquear X a menos que ya posea un bloqueo de lectura compartido o escritura para X.
- En algunas ocasiones es posible la conversión de bloqueos, lo que permite que un bloqueo de lectura pase a escritura o vice versa. Para ello, la tabla de bloqueos debe incluir identificación de transacciones en el registro de bloqueo y así saber qué transacciones poseen bloqueos.

Bloquear lectura

Data: An element X of the database
Result: Update of variable BLOCK(X)
B: **if** $BLOCK(X) = 0$ **then**
    $BLOCK(X) \leftarrow 1$;
    num_read $\leftarrow 1$;
else
    if $BLOCK(X) = 1$ **then**
        num_read $\leftarrow$ num_read $+ 1$;
    else
        wait(until $BLOCK(X) = 0$ and DBMS wakes up the transaction);
        go to B;
    end
end

Bloquear escritura

Data: An element X of the database
Result: Update of variable BLOCK(X)
B: **if** $BLOCK(X) = 0$ **then**
    $BLOCK(X) \leftarrow 2$;
else
    wait(until $BLOCK(X) = 0$ and DBMS wakes up the transaction);
    go to B;
end

Desbloquear

Data: An element X of the database
Result: Update of variable BLOCK(X)
B: **if** $BLOCK(X) = 2$ **then**
    $BLOCK(X) \leftarrow 0$;
    Wake up the waiting transactions;
else
    if $BLOCK(X) = 1$ **then**
        num_read = num_read -1;
        if $num\_read = 0$ **then**
            $BLOCK(X) \leftarrow 0$;
            Wake up the waiting transactions;
        end
    end
end

Conversión de bloqueos

Una transacción que ya tiene un bloqueo sobre x, bajo ciertas condiciones, convierte un bloqueo de un estado a otro. Por ejemplo, es posible que una transacción T emita un bloqueo compartido y más adelante lo promueva emitiendo un bloqueo exclusivo. Otra posibilidad es que una transacción puede emitir un X(x) (bloqueo exclusivo) y luego lo degrade al emitir un S(x).

Bloqueo de dos bases

- El bloqueo por si solo no garantiza la serialización porque desbloqueos pueden ocurrir antes de tiempo.
- Para garantizar la serialización se usa un protocolo denominado bloqueo de dos fases.
- Se dice que una transacción sigue el protocolo de dos fases, si TODAS las operaciones de bloqueo preceden la primera operación de desbloqueo. Entonces tenemos una fase de expansión y otra de contracción.
- Si se permite la conversión en bloqueos, los cambios de promoción (lectura a escritura) ocurren en la expansión y los de degradación en la contracción.
- El bloqueo de dos fases puede limitar la concurrencia.

Tipos de bloqueos de dos fases

- El bloqueo descrito anteriormente es el básico.
- El bloqueo de dos fases conservador o estático requiere que se bloqueen todos los elementos a los que tendrá acceso antes de comenzar a ejecutarse.
- El bloqueo estricto es en la práctica el más usado.

Bloqueo de dos fases estricto

- Cada transacción debe obtener un bloqueo compartido antes de leer X y un bloqueo exclusivo antes de escribir X
- Una transacción no libera nungún bloqueo exclusivo antes de confirmar o abortar.
- Si una T mantiene un bloqueo exclusivo sobre un objeto O, ninguna otra transacción puede tener un bloqueo compartido o exclusivo sobre O.
- Una transacción no puede pedir bloqueos adicionales sobre O una vez que libera un bloqueo de O. Bloqueo de 2 Fases estricto permite sólo planes cuyo grafo de precedencia es acíclico.
- Una variación del bloqueo de dos fases estricto es el riguroso, en el cual ningún tipo de bloqueo es liberado antes de confirmar o abortar una transacción.

Deadlock

El problema de las técnicas de bloqueo es que puede producirse un interbloqueo (deadlock) o bloqueo mortal, dos o más transacciones están esperando cada una de ellas que la otra libere algún objeto antes de seguir.

Protocolo de prevención de deadlock (1)

- Bloqueo por adelantado: obliga a que las transacciones bloqueen todos los elementos que necesitan por adelantado. En caso de no poder conseguir todos esos elementos no bloquea ninguno y se queda en espera hasta volver a intentarlo.
- Ordenación: Ordena todos los elementos de la base de datos y una transacción que necesite varios de esos elementos los bloqueará en ese orden. Esto no es muy práctico.

Protocolo de prevención de deadlock (2)

Marca de tiempo transaccional: Las transacción están marcadas con el tiempo de de iniciación de ellas. Si T1 se inicia antes que T2, entonces MT(T1) $<$ MT(T2). Bajo este protocolo hay dos esquemas que evitan el bloqueo mortal. Supongamos que Ti intenta bloquear un elemento X que ya está bloqueado por Tj con un bloqueo en conflicto, entonces dos esquemas que previenen deadlock siguen una de estas reglas:

1. Esperar_morir: Si MT(Ti) $<$ MT(Tj) (Ti es más antigua que Tj), entonces Ti puede esperar, en caso contrario (Ti es más reciente que Tj) Ti se aborta (Ti muere) y se re inicia posteriormente con la misma marca de tiempo.
2. Herir_esperar: Si MT(Ti) $<$ MT(Tj), se aborta Tj (Ti hiere a Tj) y re inicia posteriormente con la misma marca de tiempo. En caso contrario (Ti es más reciente que Tj),Ti puede esperar.

Protocolo de prevención de deadlock (3)

- No-espera: Si una transacción no puede bloquear se aborta y re inicia posteriormente.
- Espera-cautelosa: Supongamos que Ti intenta bloquear un elemento X que ya está bloqueado por Tj con un bloqueo en conflicto. Si Tj no está detenida y tiene bloqueado un elemento que Ti quiere bloquear, entonces Ti se detiene, de lo contrario se aborta Ti.

Detección de deadlock

- Se controla de forma periódica si se ha producido un deadlock. Se construye un grafo en espera, cada nodo es una transacción en ejecución y un arco de una transacción Ti a Tj, en caso que Ti esté esperando un elemento que ocupa Tj. Si existe un ciclo en el grafo tenemos un deadlock. La solución es escoger transacciones víctimas y deshacerlas, hasta que desaparezca el deadlock. Cada SGBD tiene políticas diferentes para escoger víctimas.
- Otro esquema de detección es tener un tiempo limitado de espera de una transacción, el cual al ser excedido, implica que ha ocurrido un deadlock.

Técnicas de bloqueo

Los SGBD pueden bloquear:

- un campo de un registro (un atributo de una tabla)
- un registro (una tupla)
- un archivo (una tabla)
- la BD total

Esto se llama granularidad del bloqueo. Granularidad muy gruesa implica gestionar menor número de bloqueos, pero retrasa la ejecución de muchas transacciones (los objetos no se van liberando). Una granularidad muy fina, permite mayor concurrencia, pero aparecen más situaciones de deadlock que han de ser resueltas.

Modos de bloqueos: Granularidad

Se definen intenciones de bloques de lectura (compartido S) y de escritura (exclusivo X). Antes de bloquear un item, T debe declarar intención de bloqueo en todos los ancestros. Para desbloquear, vaya de lo especíıfico a lo general. Modo SIX es bloque de lectura y de intención de escritura al mismo tiempo. Para trabajar con granularidad múltiple se define una matriz de compatibilidad con los modos de bloqueos normales (S y X) y los modelos de intención. Esta tabla se presenta a continuación:

	IS	IX	S	X
IS	√	√	√	
IX	√	√		
S	√		√	
X				

Protocolo de bloqueo múltiple

Los SGBD pueden bloquear:

- Cada transacción parte de la raíz de la jerarquía.
- Para obtener un lock S o IS sobre un nodo, debe mantener IS o IX sobre el nodo padre.
- Para obtener X o IX o SIX sobre un nodo, debe mantener IX o SIX sobre el nodo padre.
- Debe liberar locks de abajo a arriba.

El protocolo es correcto en el sentido que es equivalente a poner locks en los niveles hojas de la jerarquía.

Protocolo de bloqueo múltiple: ejemplo

- Si T1 barre R, y actualiza unas cuantas tuplas: T1 obtiene un lock SIX sobre R, y luego repetidamente obtienen locks S sobre las tuplas de R, y a veces se promociona a X sobre las tuplas.
- T2 usa un índice para leer sólo parte de R: T2 obtiene un lock IS sobre R, y luego obtiene locks S sobre tuplas de R.
- T3 lee todo R: T3 obtiene un lock S sobre R o, T3 puede comportarse como T2; y puede decidir cuál leer y lockear S.

Técnica de marca de tiempo (timestamping)

- Las marcas de tiempo son identificadores únicos que se asignan a las transacciones, que se consideran como el tiempo de inicio de una transaccin. Con esta técnica no existen bloqueos. Ordena las transacciones. Se retrasan.
- En el ordenamiento por marca de tiempo, el plan es equivalente al orden en serie específico que corresponde al orden de las marcas de tiempo de las transacciones.

Algoritmo

- El algoritmo debe asegurar que, para cada elemento en conflicto en el plan, el orden en que se tiene acceso al elemento no viole el orden de serialización de las marcas de tiempo.
- Para lograrlo el algoritmo asociada a cada elemento X de la base de datos dos valores de tiempo (MT):

  (i) MT_lectura(X): la marca de tiempo de lectura del elemento X; ésta es la mayor de todas las marcas de tiempo de las transacciones que han leído con éxito el elemento X.

  (ii) MT_escritura(X): la marca de tiempo de escritura de un elemento X; que es la mayor de todas las marcas de tiempo de las transacciones que han escrito con éxito el elemento X.

Algoritmo: ordenamiento básico (1)

La transacción T emite una operación escribir_elemento(X):

- Si MT_lectura(X) $>$ MT(T) o MT_escritura(X) $>$ MT(T), entonces abortar, revertir T y rechazar operación. Esto debe hacerse porque alguna transacción más reciente con una marca de tiempo mayor que MT(T) (y por lo tanto posterior a T en el ordenamiento por marca de tiempo) ya leyó o escribió el valor del elemento X antes de que T tuviera oportunidad de escribir X, violándose así el ordenamiento por tiempo.
- Si la situación de la parte (a) no se da, entonces, ejecutar escribir_elemento(X) de T y se asigna MT(T) a MT_escritura(X).

Algoritmo: ordenamiento básico (2)

La transacción T emite una operación leer_elemento(X):

- Si MT_escritura(X) $>$ MT(T), entonces abortar, revertir T y rechazar operación. Esto debe hacerse porque alguna transacción más reciente con una marca de tiempo mayor que MT(T) (y por lo tanto posterior a T en el ordenamiento por marca de tiempo) ya escribió el valor del elemento X antes de que T tuviera oportunidad de escribir X, violándose así el ordenamiento por tiempo.
- Si MT_escritura(X) $\leq$ MT(T), entonces, ejecutar leer_elemento(X) de T y se asigna a MT_lectura(X) el mayor valor entre MT(T) y MT_lectura(X) actual.

Algoritmo: ordenamiento estricto

- Garantiza que los planes sean tanto estrictos (para facilitar recuperación) como serializacón
- Una operación de una transacción T de escritura o lectura de X tal que MT(T) $>$ MT_escritura(X) sufre un retraso hasta que la transaccin T' que escribió el valor de X (por lo que MT(T') $=$ MT_escritura(X)) se haya confirmado o abortado. Para esto es necesario simular el bloqueo del elemento X que ha sido escrito por la transaccin T' hasta que T' haya sido abortado o confirmado. Este algoritmo, sin embargo, no provoca bloqueos mortales.

Algoritmo: la regla de Thomas

Una variante del ordenamiento básico es la regla de escritura de Thomas, que no impone la serialización por conflictos, pero rechaza menos operaciones de escritura, modificando las verificaciones de la operación escribir_elemento(X) como sigue:

- Si MT_lectura(X) $>$ MT(T), abortar, revertir T, y rechazar la operación.
- Si MT_escritura(X) $>$ MT(T) no ejecutar la operación de escritura pero continuar procesando. Esto se debe a que alguna transacción con marca de tiempo $>$ MT(T) ya escribió el valor de X. Para ello, debemos ignorar la operación escribir_elemento(X) de T porque ésta ya está obsoleta.
- Si no ocurre ninguna de las situaciones anteriores, se ejecuta la operación escribir_elemento(X) de T y se asigna MT(T) a MT_escritura(X).

Técnica optimistas

Las transacciones acceden libremente a los elementos, y antes de finalizar se determina si ha habido interferencias. Este tipo de técnicas considera que las transacciones tienen 3 fases:

- Lectura: las transacciones realizan operaciones sobre copias privadas de los objetos (accesibles solo por la transacción)
- Validación: en la que se comprueba si el conjunto de objetos modificados por una transacción se solapa con el conjunto de objetos modificados por alguna otra que haya hecho la validación durante la fase de lectura de dicha transacción.
- Grabación: en el caso de no detectar interferencias se graban las modificaciones, convirtiendo las versiones privadas de los objetos en versiones actuales.

Integridad

- La integridad tiene como función proteger la BD contra operaciones que introduzcan inconsistencias en los datos. Se habla de integridad en el sentido de corrección, validez o precisión de los datos.
- El subsistema de integridad de un SGBD debe por tanto detectar y corregir, en la medida de lo posible, las operaciones incorrectas. En la práctica es el punto débil de los SGBD comerciales, ya que casi toda la verificación de integridad se realiza mediante código de procedimientos escritos por los usuarios.
- Habrá operaciones cuya falta de corrección no sea detectado, por ejemplo, introducir un fecha de nacimiento 25/12/1945 cuando en realidad era 25/12/1954

Reglas de integridad

En general, una regla de integridad está compuesta por tres componentes:

- La restricción propiamente tal, que establece la condición que deben cumplir los datos.
- La respuesta a la transgresión, que especifica las acciones a tomar, como rechazar las operaciones, informar al usuario, corregir el error con acciones complementarias, etc.
- Condición de disparo, que especifica cuándo debe desencadenarse la acción especificada en la restricción de integridad: antes, después o durante cierto evento.

Seguridad (1)

El objetivo es proteger la BD contra accesos no autorizados. Se llama también privacidad. Incluye aspectos de:

- Aspectos legales, sociales y éticos.
- Políticas de la empresa, niveles de información pública y privada.
- Controles de tipo físico, acceso a las instalaciones.
- Identificación de usuarios: voz, retina del ojo, etc.
- Controles de sistema operativo.

Seguridad (2)

En relación al SGBD, debe mantener información de los usuarios, su tipo y los accesos y operaciones permitidas a estos. Tipos de usuarios:

- DBA, están permitidas todas las operaciones, conceder privilegios y establecer usuarios.
- Usuario con derecho a crear, borrar y modificar objetos y que además puede conceder privilegios a otros usuarios sobre los objetos que ha creado.
- Usuario con derecho a consultar, o actualizar, y sin derecho a crear o borrar objetos

Seguridad (3)

- Privilegios sobre los objetos, añadir nuevos campos, indexar, alterar la estructura de los objetos, etc. Los SGBD tienen opciones que permiten manejar la seguridad, tal como GRANT, REVOKE, etc. También tienen un archivo de auditoría en donde se registran las operaciones que realizan los usuarios.
- Otro mecanismo de seguridad que ofrecen los SGBD en entregar información a los usuarios a través de vistas (CREATE VIEW).