

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

ELEMENTOS DE BASES DE DATOS

Segundo Cuatrimestre 2014

Clase 17: Gestión de Deadlock – Control de Concurrencia y Aislamiento

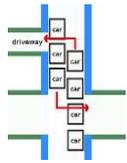


Mg. María Mercedes Vitturini
 [mvitturi@uns.edu.ar]

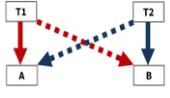
| Protocolo | Basado | ¿Deadlock? | Característica |
|-------------------------------|------------|------------|---|
| Bloqueo de Dos Fases (B2F) | Bloqueos | Si | |
| B2F Estricto | Bloqueos | Si | Mantiene locks-x hasta el final de la transacción |
| B2F Riguroso | Bloqueos | Si | Mantiene locks-s y locks-x hasta el final de la transacción |
| B2F Refinado | Bloqueos | Si | Incluye upgrade y downgrade |
| Árbol | Bloqueos | No | Solo locks-x |
| Protocolo de Estampilla (PHE) | Estampilla | No | Ordena por hora de entrada. |
| PHE + Regla de Thomas | Estampilla | No | Evita retrocesos por escrituras obsoletas |
| Validación | Estampilla | No | Protocolo optimista. Demora los controles |
| Multiversión | Estampilla | No | Las transacciones de lectura nunca retroceden |

EBD2014_17 - Mg. Mercedes Vitturini

Gestión de Deadlock



Estrategias para administrar deadlocks



Deadlock

DEADLOCK O INTERBLOQUEO: es un estado indeseable que se puede alcanzar bajo ciertos de los protocolos de control de concurrencia que incluyen bloqueos y esperas.

- “Una transacción que necesita acceder a un dato Q solicita el lock al gestor de control de concurrencia. Este se lo concederá únicamente si su *solicitud es compatible* con los locks asignados a otras transacciones sobre el mismo dato Q. Si la solicitud no es compatible, *la transacción deberá esperar.*”
- Puede darse que varias transacciones se esperen unas a otras sin que ninguna pueda seguir: *deadlock*.
- ¿Qué responsabilidad tiene el DBMS?
 - prevenir estas situaciones, ó
 - dejar que ocurran y *accionar para superarlas*.

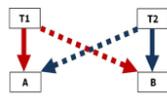


EBD2014_17 - Mg. Mercedes Vitturini

Interbloqueo o Deadlock

Consideremos las siguientes transacciones:
 T_1 : write (A); write (B);
 T_2 : write (B); write (A);
 y la planificación concurrente:

| | |
|--|---|
| T_1 Lock-X (A) Write (A) <i>Espera por Lock-X (B)</i> | T_2 Lock-X (B) Write(B) <i>Espera por Lock-X (A)</i> |
|--|---|



EBD2014_17 - Mg. Mercedes Vitturini

Interbloqueo o Deadlock

Definición: un sistema está en estado de *deadlock* si existen dos o más transacciones, tal que *toda transacción del conjunto está esperando por un dato que tiene retenido otra transacción del conjunto*.

- Ninguna transacción del grupo puede progresar y es posible que se sumen nuevas transacciones.
- Los **protocolos de bloqueos de dos fases** vistos y que no imponen orden sobre los datos tienen posibilidades de generar situaciones de deadlock.

EBD2014_17 - Mg. Mercedes Vitturini

Manejo de Deadlock

- Dos estrategias:
 - **Prevención (política pesimista):** asegura que el sistema nunca genere un estado de deadlock.
 - **Detección (política optimista):** permite que el sistema eventualmente alcance una situación de deadlock. Periódicamente se corren procesos para verificar si se produjo esta situación y corregirla.

EBD2014_17 - Mg. Mercedes Vitturini

Prevención de Deadlock

- **Estrategia #1:** Exigir a la transacción que obtenga los locks de todos los ítems de datos que requiera antes iniciar su ejecución o hacerla retroceder. Se eliminan las esperas reteniendo recursos.
 - Antes de que comience cada transacción, es difícil predecir que ítems va a requerir.
 - Limita la utilización de recursos.
 - Tiene problemas de inanición.
- **Estrategia #2:** Imponer un orden parcial sobre los ítems de datos y exigir a las transacciones que respeten el orden para obtener los mismos. Esta estrategia es usada por el protocolo de árbol visto.

EBD2014_17 - Mg. Mercedes Vitturini

Prevención de Deadlock con Estampillas

Estrategia #3: combinar locks y estampillas de tiempo.

- A cada transacción se le asigna una **estampilla de tiempo única** la primera vez que ingresa al sistema.
- La transacción que solicita un lock sobre un ítem de dato Q y que potencialmente pueden generar deadlock **espera o retrocede** según una de estas dos políticas:
 - **Wait-Die:** Si T_i requiere un dato que tiene T_j , T_i espera si $ts(T_i) < ts(T_j)$ (T_i es más antigua que T_j). De lo contrario, T_i es retrocedida.
 - **Wound-Wait:** Si T_i requiere un dato que tiene T_j , T_i espera si $ts(T_i) > ts(T_j)$ (T_i es más joven que T_j). De lo contrario, T_j es retrocedida y sus recursos apropiados por T_i .

EBD2014_17 - Mg. Mercedes Vitturini

Estrategias de Prevención con Estampillas

Wait-Die – Espera la más antigua

- (-) Una transacción puede tener que esperar varias unidades de tiempo para conseguir un recurso.

Wound-Wait – Retrocede la más joven

- (+) Puede tener menos retrocesos que wait-die.
- (+) Las transacciones más viejas tienen precedencia sobre las nuevas y de esta manera se evitan problemas de inanición.

- Tanto en el esquema **wait-die** como en **wound-wait**, una transacción que retrocede reingresa con su **estampilla de tiempo original**.

EBD2014_17 - Mg. Mercedes Vitturini

Estrategias de Prevención con Timeout

- **Estrategia #4:** basado en **timeout**. Una transacción espera por obtener un lock hasta cierta cantidad de tiempo predeterminado. Transcurrido el tiempo máximo, la transacción se debe retroceder.
 - (+) Esta política previene deadlocks.
 - (+) Es simple de implementar;
 - (-) Tiene posibilidades de inanición.
 - (-) Es difícil determinar un valor de timeout adecuado.

EBD2014_17 - Mg. Mercedes Vitturini

Deadlock: detección + Recuperación

- No usa ninguna estrategia que evite deadlocks.
- Periódicamente, el sistema controla si existe una **situación de deadlock**. Si la encuentra inicia un proceso de recuperación.
- Para esto el sistema necesita:
 1. **Mantener información** acerca de la asignación de locks sobre ítems de datos a las transacciones así como de las solicitudes demoradas.
 2. **Proveer un algoritmo de detección** para determinar si el sistema está en deadlock.
 3. **Romper el estado de deadlock** cuando el sistema detecta que existe tal situación.

EBD2014_17 - Mg. Mercedes Vitturini

Detección + Recuperación

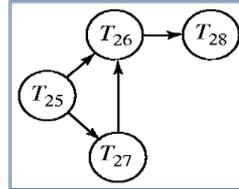
Mantener información

- Mantener un grafo de espera $G=(V, A)$
 - V (vértices) representan las transacciones del sistema.
 - A (arcos) es un par ordenado $(T_i; T_j)$. Si existe $T_i \rightarrow T_j$ en G , significa que T_i espera que T_j libere un recurso.
- Si T_j solicita un recurso que tiene T_i y que por diferencia de compatibilidades debe esperar que lo libere, se agrega $T_i \rightarrow T_j$.
- Cuando T_i libera el recurso, el arco se remueve.
- El sistema está en deadlock si en el grafo hay ciclos.**

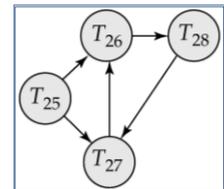
EBD2014_17 - Mg. Mercedes Vitturini

Detección + Recuperación

Detectar estado de Deadlock



Grafo de espera sin ciclos



Grafo de espera con ciclos
 ⇒ **deadlock**

EBD2014_17 - Mg. Mercedes Vitturini

Detección + Recuperación

Recuperar (romper el deadlock):

- Seleccionar una víctima:** dado un conjunto de transacciones en deadlock, determinar cuál/es transacciones deben retroceder.
- Retroceso:** determinar si una transacción debe ser retrocedida por completo o parcialmente (esto requiere información adicional del sistema).
- Inanición:** el sistema debe elegir como víctima a una transacción T_i un número finito de veces para evitar que entre en estado de inanición.

EBD2014_17 - Mg. Mercedes Vitturini

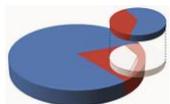
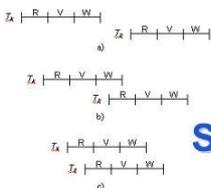
Inanición

Inanición (starvation) – define la condición por la que una transacción es demorada indefinidamente a acceder a el/los recursos que requiere porque siempre se le da preferencia a otras.

- Sea un recurso Q , que es requerido con frecuencia por varias transacciones, y una transacción T_i que requiere acceso a dicho.
- Con protocolos de bloqueo, no consigue el permiso requerido (aunque no exista situación de interbloqueo).
- Entra en situación de deadlock y es elegida como víctima.
- Con protocolos de estampilla, no cumple las condiciones de estampillas y es obligada a reingresar.
- La condición de inanición de un protocolo se complementa siguiendo alguna política para administrar inanición

EBD2014_17 - Mg. Mercedes Vitturini

Protocolo de Control de Concurrency con Bloqueos



Granularidad Múltiple

La granularidad para "Q"

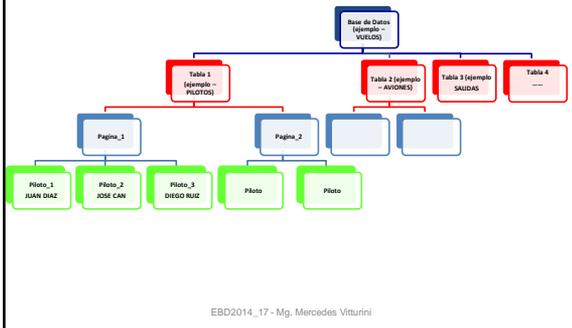
Granularidad de bloqueo (Lock-X/Lock-S (Q)) – cuál es la 'unidad' para los ítems de dato: *la base de datos, la tabla, una página o un registro.*

Depende!

- Granularidad fina (hojas del árbol):** (+) alta concurrencia, (-) alta sobrecarga (overhead) por bloqueos.
- Granularidad gruesa (niveles superiores):** (+) menor overhead por bloqueos, (-) concurrencia baja.
- Generalmente se usa granularidad fina (de tupla o registro).
- Los DBMS permiten al "elegir" la granularidad de bloqueo.

EBD2014_17 - Mg. Mercedes Vitturini

Jerarquía



EBD2014_17 - Mg. Mercedes Vitturini

Niveles de Granularidad

- **(+) Ventajas de bloqueo a nivel de registro:**
 - Menor posibilidad de conflictos si las transacciones acceden a filas distintas.
 - Se puede mantener un lock por más tiempo sin demasiado riesgo de deadlock.
- **(-) Desventajas de bloqueo a nivel de registro:**
 - La tabla de bloqueos requiere más memoria.
 - Es más lento si una operación requiere de varios bloqueos, por ejemplo GROUP BY, se debe conseguir el lock de cada registro
- **Bloqueos a mayor nivel (tabla o página) son apropiados si:**
 - La mayoría de las operaciones son de lectura.
 - Muchos accesos usando consultas GROUP BY.

EBD2014_17 - Mg. Mercedes Vitturini

Protocolo con Granularidad Múltiple

Bloqueos con granularidad múltiple – protocolo que permite que coexistan bloqueos de datos en unidades distintas para diversas transacciones, esto es, una transacción decide individualmente el nodo del árbol para el que requiere un lock.

Ejemplo:

- Una transacción T_1 requiere un lock-s sobre la relación (o tabla) **CLIENTES**, mientras que otra transacción T_2 requiere un lock-X a nivel de registro para el cliente ID= 00123.
- El modo de lock (hasta ahora *exclusivo* o *compartido*) alcanza al nodo de la jerarquía y sus descendientes, si existen.
- Para hacer práctica la gestión de bloqueos con granularidad múltiple, se emplean tipos adicionales de bloqueo, llamados **locks de intención (intention locks)**.

EBD2014_17 - Mg. Mercedes Vitturini

El gestor de bloqueos

- Algunos posibles niveles para un lock: base de datos, tabla, página o bloque, registro
 - Tener un lock en un nivel de la jerarquía implica tener un lock del mismo modo sobre los descendientes.
 - Una transacción T_1 que tiene un lock-s sobre la relación (o tabla) **CLIENTES**, implícitamente también tiene un lock-s sobre cada registro de la tabla.
 - **Problema!** el Gestor de Bloqueos debe asegurar permisos consistentes, aún cuando existen **locks implícitos**:
 - Ejemplo:
 - T_1 tiene un lock-x sobre un registro de la tabla y T_2 requiere lock-s sobre la tabla.
- ¿cómo determinar si se puede conceder o no un lock?**

EBD2014_17 - Mg. Mercedes Vitturini

Intención de Bloqueos

Solución

- Se introducen nuevos modos: **modo lock con intención (intention lock mode)**
 - **IS (intention-share mode)**: en algún nivel inferior existen bloqueos explícitos compartidos.
 - **IX (intention-exclusive mode)**: en algún nivel inferior existen bloqueos explícitos exclusivos o compartidos.
 - **SIX (shared, intention-exclusive mode)**: el subárbol está bloqueado explícitamente en modo compartido y algún hijo en modo exclusivo.
- El problema del Gestor de Bloqueos es cómo asegurar permisos consistentes cuando existen **bloqueos implícitos**:
¿cómo determinar si se puede conceder o no un lock?

EBD2014_17 - Mg. Mercedes Vitturini

Matriz de Compatibilidades

- Las compatibilidades para todas las posibilidades de locks es :

| | IS | IX | S | SIX | X |
|-----|----|----|---|-----|---|
| IS | ✓ | ✓ | ✓ | ✓ | × |
| IX | ✓ | ✓ | × | × | × |
| S | ✓ | × | ✓ | × | × |
| SIX | ✓ | × | × | × | × |
| X | × | × | × | × | × |

Y existe (circulo en X)

pedido (circulo en IX)

EBD2014_17 - Mg. Mercedes Vitturini

Protocolo de Bloqueo con Granularidad Múltiple

Para que T_i consiga bloquear un nodo Q del árbol:

- Respetar la matriz de compatibilidad.
- Bloquear primero la raíz en algún modo (puede ser en cualquier modo).
- Para obtener un **lock sobre un nodo Q en modo S o IS**, actualmente debe tener el lock sobre al padre de Q en IS o IX.
- Para obtener un **lock sobre un nodo Q en modo X, IX o SIX** actualmente debe tener el lock sobre el padre de Q en IX o SIX.
- No se puede conseguir un nuevo bloqueo si ya se libero alguno (dos fases).
- Solo se puede liberar un lock si ya libero los locks de los hijos.

EBD2014_17 - Mg. Mercedes Vitturini

Ejemplos

- **T1 necesita leer y mostrar el domicilio del piloto Diego Ruiz (Lock-s a nivel de registro).**
 - T1 debe conseguir IS sobre la base de datos, la tabla PILOTOS, la página_1 de Pilotos y S sobre el registro de Diego Ruiz.
- **T2 actualiza las horas de vuelo del piloto Juan Roldán (requiere Lock-X a nivel de registro).**
 - T2 necesita IX sobre la base de datos, la tabla PILOTOS, la página_1 de Pilotos y X sobre el registro de Juan Roldán.
- **T3 consulta las horas de vuelo de todos los pilotos y actualiza las horas de vuelo de los que viajaron la última semana (requiere lock-S-IX a nivel de tabla).**
 - T3 necesita IX sobre la base de datos y SIX sobre la tabla pilotos y X sobre los registros pilotos que cumplen con la condición.

EBD2014_17 - Mg. Mercedes Vitturini

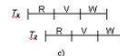
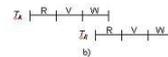
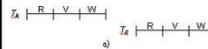
Protocolo Granularidad-Multiple

Propiedades del Protocolo

- El protocolo de bloqueos de granularidad múltiple con dos fases garantiza *planificaciones serializables*.
- *No está libre de deadlock.*

EBD2014_17 - Mg. Mercedes Vitturini

Implementaciones en Control de Concurrencia



Multiversión + 2PL



Multiversión + 2PL

- Diferencia entre **transacciones sólo de lectura (TrR)** y **transacciones de actualización (TrW)**.
- Las **TrW** deben adquirir el lock (S ó X) y retenerlos hasta que la transacción termine (dos fases)
 - Cada escritura exitosa crea una nueva versión del ítem de datos.
 - Cada versión de un ítem de dato tiene una estampilla de tiempo cuyo valor se obtiene de un **ts-counter** que se incrementa con el proceso de commit de la transacción.
- A las **TrR** se les asigna una estampilla de tiempo que corresponde con el valor actual de **ts-counter** antes de iniciar su ejecución. Las lecturas siguen el protocolo multiversión

EBD2014_17 - Mg. Mercedes Vitturini

Multiversión + 2PL

- Si TrW_i requiere $read(Q)$, debe conseguir un $lock-S(Q)$ sobre la última versión de Q.
- Si TrW_i requiere un $write(Q)$, debe obtener un $lock-X(Q)$ sobre la última versión de Q y crear la nueva versión con estampilla de tiempo infinito.
- Cuando TrW_i se completa, el proceso de commit:
 - Asigna a TrW_i la estampilla de tiempo $ts-counter + 1$ y guarda el nuevo valor de $ts-counter$.
- Las TrR_j que comiencen después de que TrW_i cometió verán el valor actualizado.
- Las TrR_j que comiencen antes de que TrW_i cometan verán el valor anterior.

EBD2014_17 - Mg. Mercedes Vitturini

Niveles de Aislamiento de SQL'92

- SQL'92 define cuatro niveles de aislamiento. Los menos estrictos **pueden generar ejecuciones no serializables:**
Serializable
Repeatable read – autoriza la lectura de registros que pertenecen a transacciones cometidas. Si la misma transacción necesita leer el mismo registro varias veces se devuelve el mismo valor.
Read committed – autoriza la lectura de registros que pertenecen a transacciones cometidas.
Read uncommitted – permite leer datos de generados por transacciones no cometidas.

EBD2014_17 - Mg. Mercedes Vitturini

Niveles de Aislamiento

| Nivel | Dirty Reads | NonRepeatable Reads | Phantom |
|-------------------------|-------------|---------------------|---------|
| READ UNCOMMITTED | ✓ | ✓ | ✓ |
| READ COMMITTED | - | ✓ | ✓ |
| REPEATABLE READ | - | - | ✓ |
| SERIALIZABLE | - | - | - |

EBD2014_17 - Mg. Mercedes Vitturini

Niveles de Aislamiento

- Los DBMS definen un nivel de consistencia por defecto, generalmente **read committed**.
- Es posible explícitamente cambiarlo:
 - Modificar los parámetros del sistema.
 - Modificar las condiciones para una transacción en particular con la instrucción SQL
 - **SET ISOLATION LEVEL ...**

EBD2014_17 - Mg. Mercedes Vitturini

Bloqueos a Nivel de Índice

- Index locking protocols** – provee concurrencia y previene **phantom** fijando los locks en el registro índice, en lugar de sobre los datos.
- Todas las relaciones o tablas deben tener por los menos un índice y el acceso a las tuplas debe ser a través del índice, que se bloquea con lock-S
 - Una transacción que inserta una tupla requiere escribir el índice lock-X.
 - Puede prevenir el fenómeno **phantom**.

EBD2014_17 - Mg. Mercedes Vitturini

Temas de la clase de hoy

- Protocolos de control de concurrencia:
 - Granularidad Múltiple.
 - Multiversión y 2PL
- Gestión de deadlocks
 - Prevención.
 - Detección
- Bibliografía**
 - “Database System Concepts” – A. Silberschatz. Capítulo 16 y 26 a 29.

EBD2014_17 - Mg. Mercedes Vitturini