Introducción a las Bases de Datos TUPAR

Cursada 2008

Clase 8: Ambiente de BD Seguridad, Recuperación y Concurrencia

Facultad de Ciencias Exactas
Universidad Nac. Centro de la Pcia. de Bs. As.

Seguridad

Seguridad

Seguridad en las bases de datos → → protección contra:

- Revelación no autorizada (confidencialidad)
- Alteración no autorizada (integridad)
- Destrucción intencional o involuntaria

Protección dirigida a dos tipos de usuarios →

- Los que no tienen derechos de acceso
- Los que tienen derechos limitados a ciertas acciones

Seguridad ≡ ¿Que datos? + ¿Que operaciones? + ¿Cuales usuarios?

Seguridad en Bases de Datos

- Datos → activo más valioso de una organización
  → necesidad de controlarlo y administrarlo cuidadosamente.
- Parte o la totalidad de los datos corporativos
  → importancia estratégica
  → necesidad de manejarlos en forma segura y confidencial.
- Consideraciones sobre seguridad
  → no se aplican sólo a los datos
  → brechas en la seguridad pueden afectar otras partes del sistema, que a su vez pueden afectar la BD.

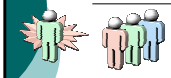
Seguridad en Bases de Datos

- Cuán valiosos son los datos? Cuáles datos deben asegurarse?
  Diferentes tipos de datos → diferentes niveles de seguridad.
  Datos públicos no requieren el mismo nivel de seguridad que datos privados.
- Cuánto costarían los accesos ilegales a los datos?
  Si una porción de los datos tiene mucho valor para la organización, el acceso ilegal puede ser muy perjudicial. El 'costo' de 'perder' los datos determina la calidad de la seguridad requerida.

Seguridad en Bases de Datos

- Cuáles son las implicaciones de los cambios o destrucciones de datos?
  Si la pérdida de datos produce consecuencias desastrosas entonces los niveles de seguridad deben ser altos.
- Las medidas de seguridad afectarán el funcionamiento de la base de datos?
  No serán útiles los sistemas de seguridad que impidan en cualquier caso el acceso a los datos a un usuario legítimo.

Seguridad en Bases de Datos



Atacantes vs. Defensores

El Atacante necesita conocer sólo UN punto de vulnerabilidad pero … el Defensor necesita asegurar TODOS los puntos de entrada !!!

Atacantes tienen todo el tiempo del mundo pero … el Defensor trabaja con restricciones de tiempo y costos !!!

Seguridad vs. Usabilidad

Sistemas seguros son más difíciles de usar

P.Ej. Passwords complejas y muy elaboradas son difíciles de recordar

Securidad como una idea tardía

Desarrolladores y Administradores pueden pensar que la seguridad no añade ningún valor comercial/organizacional

Ocuparse de las vulnerabilidaddes justo antes de que un producto se comercialice puede ser caro pero …

Aspectos Relativos a Seguridad

- Morales/Éticos → puede haber razones morales que regulen quienes tienen acceso a determinada información, por ejemplo registros médicos, de antecedentes penales, etc.

- Requisitos Legales → se requiere que las organizaciones usen con discreción los datos personales de los individuos. Esta información debería poder ser verificada por los individuos mismos.

Aspectos Relativos a Seguridad

- Seguridad Comercial → Información perteneciente a una empresa es un recurso valioso que podría ser útil para la competencia.
- Fraude/Sabotaje → La información podría ser mal utilizada, por ej. por un novato o por alguien que tiene la intención de confundir.
- Errores → cambios accidentales en los datos, no maliciosos.

Elementos que pueden ser protegidos

Granularidad

- Un atributo de una tupla.
- Un conjunto de columnas.
- Una tupla individual.
- Un conjunto de tuplas de una relación.
- Una relación en particular.
- Un conjunto de relaciones.
- La base de datos completa

Métodos para el Control de Accesos

- Control de Acceso Discrecional
  garantiza privilegios a usuarios, incluyendo la capacidad para acceder archivos de datos específicos, registros o campos para operar de una manera determinada (read, insert, delete, o update).

- Control de Acceso Mandatorio
  clasifica usuarios y datos en múltiples niveles de seguridad, y luego fuerza determinadas reglas acordes a cada nivel.

Seguridad a Nivel de Usuario en SQL

Cada usuario tiene ciertos derechos sobre ciertos objetos.

Distintos usuarios → los mismos o distintos derechos sobre los mismos objetos.

Para controlar la granularidad de los derechos de acceso, los usuarios pueden tener derechos (autorización / privilegios) sobre

- Tabla
- Vista → controla particiones horizontales (selecciones) y verticales (proyecciones) de una tabla y datos generados dinámicamente desde otras tablas.

Control de Acceso Discrecional

GRANT SCHEMA NbreEsqBD AUTHORIZATION usuario;

GRANT privilegios ON objeto TO usuarios [WITH GRANT OPTION]

REVOKE [GRANT OPTION FOR] privilegio ON objeto FROM usuarios {CASCADE | RESTRICT}

Control de Acceso Discrecional

Privilegios a asignar:

- SELECT – para leer todas las columnas (incluyendo las que se añadan con ALTER TABLE)
- DELETE – para remover datos
- INSERT (columna/s) – para incorporar nuevas tuplas con valores no nulos (o no default) en esa/s columna/s.
- INSERT ídem para todas las columnas.
- UPDATE – análogo a INSERT para modificar datos existentes
- REFERENCES (columna) - para definir : foreign keys (en otras tablas) que referencien a esta columna.
- Sólo el propietario puede ejecutar CREATE, ALTER, y DROP.

With Grant Option

WITH GRANT OPTION permite que el poseedor de ciertos privilegios pueda transmitirlos a otros usuarios.

Usuario → puede ser un 'username' o PUBLIC

PUBLIC → los privilegios se asignan a todos (ej. GRANT SELECT ON ListaAlumnos TO PUBLIC;)

Control de Acceso Discrecional: Ejemplo

- GRANT INSERT, SELECT ON Atletas TO Homero

Homero puede insertar y seleccionar tuplas de Atletas

- GRANT DELETE ON Atletas TO Entrenador WITH GRANT OPTION
- Entrenador puede borrar tuplas de Atletas y autorizar borrados a otros usuarios.
- GRANT UPDATE (categoría) ON Atletas TO Organizador
- Organizador puede actualizar solamente la categoría en las tuplas de Atletas.
- GRANT SELECT ON VistaAtletasVeteranos TO Juan, Ivan, Ines
- Juan, Ivan e Ines NO pueden consultar directamente la tabla Atletas!
- REVOKE: cuando un privilegio le es revocado al usuarioX, también le es revocado a los que lo obtuvieron solamente de usuarioX.

Control de Acceso Discrecional

GRANT/REVOKE en Vistas

- El creador de una vista tiene privilegios sobre la vista si los tiene sobre todas las tablas subyacentes.
- Si el creador de una vista pierde un privilegio obtenido con With Grant Option, sobre una tabla subyacente, también pierde el privilegio sobre la vista; lo mismo ocurre con los demás usuarios que obtuvieron el privilegio sobre la vista!

REVOKE

José: GRANT SELECT ON Atletas TO Martin WITH GRANT OPTION

Martín: GRANT SELECT ON Atletas TO Juan WITH GRANT OPTION

José: REVOKE SELECT ON Atletas FROM Martin CASCADE

CASCADE vs RESTRICT:

- CASCADE: Todos los privilegios 'colgados' también son revocados
- RESTRICT: El comando REVOKE es rechazado si produce privilegios 'colgados'.

REVOKE

José: GRANT SELECT ON Atletas TO Martin WITH GRANT OPTION

José: GRANT SELECT ON Atletas TO Juan WITH GRANT OPTION

Martín: GRANT SELECT ON Atletas TO Juan WITH GRANT OPTION

José: REVOKE SELECT ON Atletas FROM Martin CASCADE

Juan retiene sus privilegios.

REVOKE

José: GRANT SELECT ON Atletas TO Martin WITH GRANT OPTION

José: GRANT SELECT ON Atletas TO Martin WITH GRANT OPTION

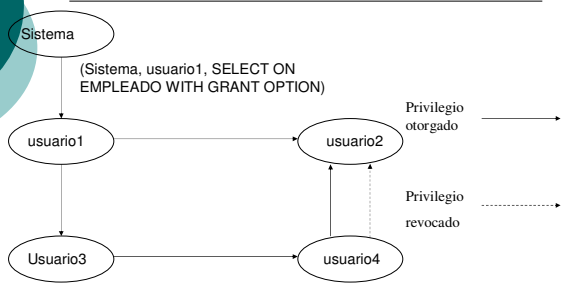
José: REVOKE SELECT ON Atletas FROM Martin CASCADE

Los privilegios de Martin son revocados.

Nota: se podría haber revocado GRANT OPTION:

- REVOKE GRANT OPTION ON Atletas FROM Martin CASCADE

Debilidades del Sistema Discrecional



Grafo de Autorizaciones → usuario 2 conserva los privilegios

Sistema Discrecional: Resumen

- El acceso al esquema (colección de objetos asociados con un usuario: tablas, vistas, índices, etc.) → otorgada a otros usuarios 'a discreción' del usuario poseedor.
- Usuarios: nombres que la BD reconoce como autorizados para acceder a la BD.
- Privilegios: derechos para acceder o ejecutar un procedimiento SQL sobre datos de otro usuario → diferentes tipoos de privilegios: conexión, creación de tablas, etc.
- GRANT – REVOKE

Roles

- Problema

→ muchos usuarios con muchos privilegios diferentes
→ difícil adicionar nuevos privilegios a cada individuo
→ **ROLES**

- Grupos de privilegios relacionados que se otorgan a usuarios.
- Si se cambian los privilegios encapsulados en un rol, los privilegios de todos los usuarios que tienen ese rol también cambian !!

Autorización Basada en Roles

- En SQL-92, los privilegios son asignados realmente a identificadores que pueden representar a un usuario aislado o a un grupo de ellos.
- En SQL:1999 (y en muchos sistemas actuales) los privilegios son asignados a roles.

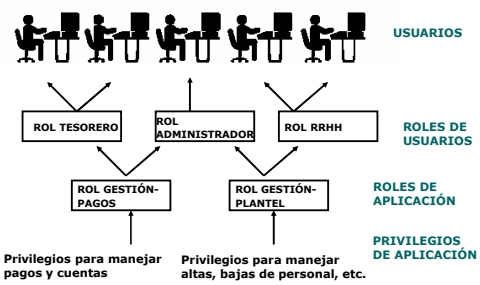
```
CREATE ROLE <nombre rol> [ WITH ADMIN <quien> ]

GRANT <rol concedido> [ { , <rol concedido> }... ] TO <a-quien> [ { , <a-quien> }... ]
      [ WITH ADMIN OPTION ] [ GRANTED BY <quien> ]
```

- Roles pueden ser asignados a usuarios o a otros roles <a-quien>.
- Es una disposición más cercana a la forma en que trabajan las organizaciones.

Autorización Basada en Roles

Roles pueden ser asignados a usuarios o a otros roles <a-quien>.



Guías de Seguridad para el DBMS

Usar la menor funcionalidad posible

- La menor cantidad de protocolos de comunicación que sea posible
- Borrar todos los procedimientos del sistema que sean innecesarios o inútiles.
- Desabilitar login por defecto y usuarios invitados hasta donde sea posible.
- A menos que se requiera no permitir a todos los usuarios que se logueen con el DBMS interactivamente.

Guías de Seguridad para el DBMS

- Correr el DBMS detrás de un firewall (dispositivo que se utiliza en redes de computadoras para controlar las comunicaciones, permitiéndolas o denegándolas), pero planificar todo como si el firewall no estuviese activo.
- Utilizar las últimas versiones del SO y DBMS y sus packs.
- Proteger la computadora sobre la que corre el DBMS
  - No permitir usuarios que estén trabajando en la computadora que tiene el DBMS
  - Tratar de ubicar la computadora que tiene el DBMS en ambientes físicamente seguros
  - El acceso a esta habitación debería ser registrado en un log (diario de registro)

Guías de Seguridad para el DBMS

- Manejar cuentas y passwords
  - Usuarios con pocos privilegios para el servicio del DBMS
  - Proteger las cuentas de la BD con passwords MUY seguras
  - Auditoría de intentos fallidos a la BD
  - Chequeo de grupos y usuarios frecuentemente
  - Auditoría de cuentas sin passwords
  - Asignar a cada grupo la menor cantidad posible de privilegios
  - Limitar los privilegios para la cuenta del DBA
- Planeamiento
  - Desarrollar un plan de seguridad para prevenir y detectar problemas
  - Crear procedimientos/protocolos para emergencias de seguridad y practicarlos !!

DBMSs y Seguridad en la Web

- Se debe asegurar que la información transmitida:
  - sea inaccesible salvo para el emisor y el receptor (privacidad).
  - No cambie durante la transmisión (integridad);
  - El receptor esté seguro de que proviene del emisor (autenticidad);
  - El emisor sepa que receptor es genuino (no-fabricación);
  - El emisor no pueda negar que ha hecho el envío (no-repudio).
- Debe proteger la información una vez que ha alcanzado el servidor de WEB.

Recuperación

Introducción a la Recuperación

Una base de datos es:

Un conjunto de datos integrados , adecuado a varios usuarios y a diferentes usos

- el uso concurrente de los datos plantea problemas de seguridad que deben ser paliadas con las facilidades que e proporciona el DBMS.
- La protección de los datos deberá llevarse a cabo contra fallos físicos, fallos lógicos y fallos humanos (intencionados o no).

Introducción a la Recuperación

El DBMS facilita mecanismos para

- prevenir los fallos (subsistema de control),
- detectarlos una vez que se han producido (subsistema de detección)
- corregirlos después de haber sido detectados (subsistema de recuperación).

Significa →

- Examinar las implicaciones de los fallos en las transacciones, sobre la base de datos.
- Examinar maneras de prevenir los problemas que pueden ocurrir como resultado de fallos en el sistema.

Transacciones

Transacción: conjunto de operaciones que forman una unidad conceptual de procesamiento.

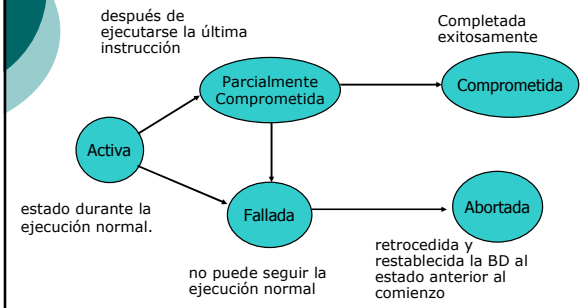
Propiedades **ACID**

- **Atomicidad**: se ejecutan todas las operaciones de la transacción o no lo hace ninguna de ellas → si algo falla no deben quedar rastros de la ejecución inconclusa en la BD → Gestor de recuperación (Recovery Manager) + Gestor de Transacciones (Transaction Manager)
- **Consistencia**: la ejecución de la transacción conserva la consistencia de la BD → Administrador de Concurrencia (Concurrency Manager) + restricciones

Transacciones

- Propiedades **ACID**
  - **Aislamiento** (isolation): cada transacción se ejecuta sin interferencia del resto de las transacciones que se ejecutan concurrentemente en el sistema → Gestor de Transacciones
  - **Durabilidad**: los cambios de una transacción finalizada exitosamente deben ser permanentes en la BD, incluso si hay fallos en el sistema → Gestor de recuperación

Diagrama de estado de una transacción



Transacciones

- Transacción abortada →
  - ACCIONES:
    - Error no vinculado a lógica de la transacción (hardware o software) → Reinicio
    - Error en la lógica del programa que ejecuta la transacción → Cancelar

Fallos con Pérdida de Información

Muchos **tipos diferentes de fallos** que pueden afectar al procesamiento de la base de datos

Cada uno debe ser tratado de forma distinta

- Paradas catastróficas del sistema debidas a errores del hardware o del software → pérdida de contenido de la memoria principal
- Fallos del soporte físico → pérdida de la información guardada en almacenamiento secundario

Fallos con Pérdida de Información

- Errores de software de las aplicaciones → fallo en una o más transacciones
- Desastres físicos naturales como incendios, inundaciones, terremotos y apagones
- Destrucción negligente o no intencionada de datos o instalaciones por operadores o usuarios
- Sabotaje o corrupción o destrucción intencionada de los datos, del hardware, del software o de las aplicaciones

Tratamientos ante Fallos

Eventos indeseables

- Esperados
  - Acciones ejecutadas durante el procesamiento normal de la transacción que permiten la recuperación ante fallos

→ integran la lógica de la transacción

Tratamientos ante Fallos

Eventos indeseables

- Inesperados
  - Acciones ejecutadas después de ocurrir el fallo para restablecer el contenido de la BD

→ garantizar las propiedades ACID

Retornar al estado consistente previo a la ejecución de la transacción fallida.

Algoritmos de Recuperación

Los algoritmos de Recuperación son técnicas que garantizan las propiedades A, C y D a pesar de las fallas.

- Tienen dos partes:
  1. Acciones durante el procesamiento normal de una transacción para asegurar que hay suficiente información de ayuda a la recuperación en caso de falla.
  2. Acciones ejecutadas después de una falla para recuperar el estado de consistencia de la base (propiedades A, C y D)

Recuperación

Un SGBD debe proporcionar las siguientes funcionalidades como ayuda a la recuperación:

- Mecanismo de copia de seguridad mediante el que se hagan copias de seguridad periódicas de la base de datos
- Facilidades de registro que mantengan el control del estado actual de las transacciones y de los cambios realizados en la base de datos
- Funcionalidad de puntos de comprobación que permita que las actualizaciones de la base de datos que estén llevándose a cabo se hagan permanentes
- Gestor de recuperación que permita al sistema restaurar la base de datos a un estado coherente después del fallo

Recuperación

Estrategias típicas
- Copias de respaldo en otros medios de almacenamiento (backups)
- Operaciones específicas: Redo (Rehacer), Undo (Deshacer)

- Medio para registrar los movimientos realizados →
  - LOG de transacciones (diario o bitácora)
- Métodos de recuperación basadas en el contenido del log →
  - Actualización diferida (Deferred update)
  - Actualización inmediata (Immediate update)
- Técnicas para recuperación →
  - Vía reprocesamiento
  - Vía rollback/rollforward (Deshacer/rehacer)

Recuperación vía Reprocesamiento

- La base de datos se retorna a un estado correcto previo conocido y se reprocesan las transacciones ejecutadas hasta el momento del fallo

→ Estrategia impráctica en algunos casos y no factible en otros →
- El sistema tiene una carga de trabajo asincrónica, eventos concurrentes, etc.

Recuperación vía Rollback/Rollforward

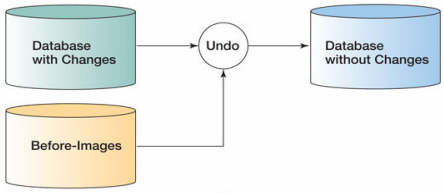
- Salvar la BD periódicamente y mantener un archivo auxiliar con los cambios producidos (**LOG**) en orden cronológico.
- Write-Ahead Logging (WAL)→ PRIMERO se escribe en el LOG
  - Fuerza las escrituras en el LOG antes de escribir en la base de datos (#1)
  - Fuerza el Commit de la transacción después de escribir en el LOG → escribe todos los registros del Log de la transacción antes del commit (#2)
- #1 garantiza Atomicidad.
- #2 garantiza Persistencia (durabilidad)

Recuperación vía Rollback/Rollforward

- Cuando ocurre un fallo →
  - Rollback: deshace los cambios erróneos y reprocesa las transacciones válidas
  - Rollforward: rehace los cambios en la base de datos usando datos salvados y transacciones válidas desde el último backup

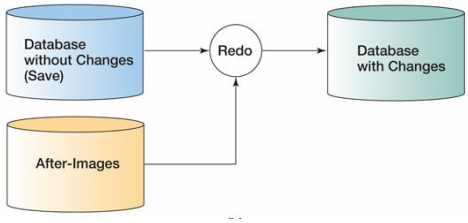
Rollback

Imagen 'antes de': una copia de cada registro de la BD (o página) antes de ser modificada.



Rollforward

Imagen 'después de' : una copia de cada registro de la BR (o página) después de haber sido cambiada



Backups

- Backups regulares salvan el estado de la base de datos en un momento determinado
  - Útil para restaurar la base de datos en caso de fallas catastróficas (destrucción física, sabotaje, etc.)
  - Operación costosa en tiempo → ejecutada cuando el sistema está ocioso o con menor carga de trabajo
  - Backup incremental → copia de respaldo frecuente de aquéllos registros que se han modificado
  - Cómo asegurar persistencia de los cambios (Durabilidad ACI**D**)

Recuperación basada en el LOG

- Cada registro de cambio del LOG contiene →
  - Identificador de la transacción (Id_T) que ejecuta el cambio
  - Identificador del ítem de datos modificado (típicamente la ubicación en el disco)
  - Valor viejo (que fue sobreescrito)
  - Valor nuevo (luego del write)
- Otros registros del LOG contienen →
  - <Id_T, tiempo de comienzo> → transacción ACTIVA
  - <Id_T, tiempo de commit>→ transacción confirmada
  - <Id_T, tiempo de aborto> → transacción abortada
- El LOG tiene los datos completos de los cambios desde el último backup
- Debe almacenarse en memoria estable !!

Recuperación basada en el LOG

- Operación de recuperación usa dos primitivas →
  - redo: rehace la actualización registrada en el LOG.
    - Escribe el valor nuevo sobre el ítem de datos modificado
  - undo: deshace la actualización registrada en el LOG
    - Escribe el valor viejo sobre el ítem de datos modificado

Recuperación basada en el LOG

- Ambas primitivas ignoran el estado del registro en la base de datos → sobreescriben directamente.
- La aplicación múltiple de estas operaciones es equivalente a la aplicación de la última vez → idempotencia de undo y redo

redo (redo (redo (…(redo(x))))) ≡ redo (x)
undo (undo (undo (…(undo(x))))) ≡ undo (x)

Checkpoints

- Cuando se produce un fallo …
  - Que acciones hay que rehacer (redo)?
  - Que acciones hay que deshacer (undo)?
- Inspección del LOG completo →
  - Costoso en tiempo
  - Muchas transacciones ya habrán confirmado sus cambios en la base de datos (ya son persistentes) → rehacerlas es inofensivo pero se pierde tiempo

Checkpoints

- Para reducir la carga de trabajo que implica la restauración → checkpoints (puntos de verificación).
  1. Grabar todos los registros del LOG actualmente en memoria principal en almacenamiento estable.
  2. Grabar en la BD todos los bloques de registros almacenados en buffers.
  3. Escribir en el LOG un registro < checkpoint>.

Checkpoints

Durante la recuperación → considerar solamente la transacción T_i más reciente comenzada antes del checkpoint, y las que comenzaron después de T_i.

1. Recorrer hacia atrás desde el final del LOG, hasta alcanzar el <**checkpoint**> más reciente.
2. Continuar recorriendo hasta encontrar <T_i **start**>.
3. Considerar la parte del LOG que sigue a este registro.

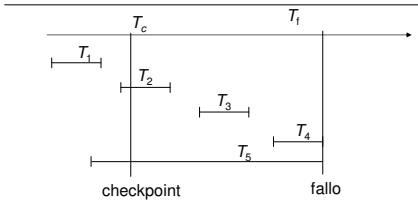
Checkpoints

1. Para todas las transacciones (comenzadas desde T_i o después) que no hayan registrado <T_i **commit**>, ejecutar undo(T_i) → *tiene sentido en actualizaciones inmediatas o de registro desconocido!!*
2. Recorrer el LOG hacia adelante para todas las transacciones comenzadas desde T_i *que hayan registrado* <T_i **commit**>, ejecutando redo(T_i).

Checkpoints

- No esperar a que las transacciones activas finalicen
- No permitir que hagan modificaciones en los buffers o en el LOG mientras dura el checkpointing
- Debe insertarse un registro de checkpointing en el LOG, que incluya la lista de transacciones activas
  → <checkpoint, L>
- Para la recuperación se necesita recorrer los registros correspondientes a las transacciones indicadas en L, para deshacer o rehacer cambios.

Esquema de Situaciones con Checkpoint



- T_1 puede ignorarse → cambios registrados en la BD por el checkpoint
- T_2 y T_3 → rehacer → **D**urabilidad
- T_4 y T_5 → deshacer → **A**tomicidad

Recuperación de Bases de Datos

El Commit para esta transacción no existe → imagen 'antes de'

T18: **start.**
T83: **start**
T18: 1982374, Balance, $900.00, $950.00
T29: **start**
T83: 3874843, Balance, $20.00, $70.00
T29: 4948543, Balance, $1350.00, $350.00
T18: 9384945, Balance, $200.00, $250.00
T18: **commit**
T29: 3984554, Balance, $900.00, $1900.00
T29: **commit**
Falla

Recuperación de Bases de Datos

El Commit de estas transacciones fue provisto → usar imagen 'despues de'

T18: **start.**
T83: **start**
T18: 1982374, Balance, $900.00, $950.00
T29: **start**
T83: 3874843, Balance, $20.00, $70.00
T29: 4948543, Balance, $1350.00, $350.00
T18: 9384945, Balance, $200.00, $250.00
T18: **commit**
T29: 3984554, Balance, $900.00, $1900.00
T29: **commit**
Falla

Recuperación basada en Act. Diferidas

- Las transacciones no cambian la BD hasta que alcanzan el punto de confirmación
  - No lo alcanzan hasta que el LOG no ha sido actualizado

→ Estrategia NoUndo/Redo

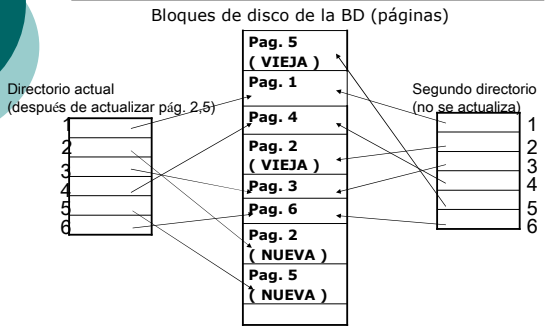
Recuperación basada en Act. Inmediatas

- Actualiza "immediatamente" la BD usando WAL
- Dos estrategias
  - Undo/NoRedo → actualiza la BD en el disco antes del commit
  - Undo/Redo → puede hacer commit antes de que los cambios se hayan grabado efectivamente en la BD (puede haber cambios en buffers)

Doble Paginación (Shadow Paging)

- Considera la BD constituida por páginas de disco de tamaño fijo
- El segundo directorio no cambia y el corriente apunta a la página más reciente
- Estrategia NoUndo/NoRedo

Doble Paginación (Shadow Paging)



Concurrencia

Control de Concurrencia

Objetivo

- Forzar el aislamiento de transacciones conflictivas mediante la exclusión mutua.
- Preservar la consistencia de la BD mediante la ejecución adecuada de las transacciones.
- Resolver los conflictos de escritura simultánea y de lecto-escritura simultáneas.

A**I**slamiento (+ **C**onsistencia) → Control de Concurrencia

Operaciones Conflictivas

- Plan de ejecución (schedule): desarrollo en el tiempo de la ejecución de transacciones concurrentes → ejecución intercalada de sentencias de las transacciones.
- Un plan mantiene el orden de las operaciones dentro de cada transacción.
- Dos operaciones de un plan de ejecución son conflictivas si satisfacen las siguientes condiciones:
  (1) Pertenecen a diferentes transacciones
  (2) Intentan acceder al mismo ítem (read-item/write-item)
  (3) Una o las dos operaciones es de escritura (write-item)

Operaciones Conflictivas

- Dos transacciones que acceden al mismo ítem intercaladas de manera que producen un resultado incorrecto.

T1	T2	X	Y
read_item(X);		40	
X ← X - N;		38	
	read_item(X);	40	
	X ← X + M;	43	
write_item(X);		38	
read_item(Y);		85	
	write_item(X);	43	
Y ← Y + N;			87
write_item(Y);			87

X=40
Y=85
N=2
M=3

- X fue sobreescrito !! Los valores correctos debieron ser:T1, T2 → X=41, Y=87 o bien T2, T1 → X=41 , Y=87

Problemas típicos de la concurrencia

Actualización perdida (Lost Update Problem)

Análisis (sumario) Inconsistente o Lectura no Repetible (Unrepeatable Read Problem)

Lectura Sucia o Fantasma (Dirty Read Problem)

Actualización Perdida

T1	T2	X = 40	Y = 5
read_item(X);		40	
read_item(Y);			5
	read_item(X);	40	
	read_item(Y);		5
X ← X – 2*Y		30	
	X ← Y;	5	
write_item(X);		30	
commit			
	write_item(X);	5	
	commit		

Trans-1, Trans-2 → X=5 Trans-2, Trans-1 → X= -5

No necesariamente las diferentes ejecuciones seriales arrojan el mismo resultado !!

Análisis Inconsistente

A=4
X=4
Y=8
N=2

T1:	T2:	T1	T2	suma
	Suma ← 0;			0
	read_item(A);		4	
	Suma ← suma + A;			4
read_item(X);		4		
X ← X - N;		2		
write_item(X);		2		
	read_item(X);		2	
	suma ← sum + X;			6
	read_item(Y);		8	
	suma ← suma + Y;			14
read_item(Y);		8		
Y ← Y + N;		10		
write_item(Y);		10		

El análisis inconsistente ocurre por utilizar registros en distinto estado de actualización!!→T2 utiliza X actualizado e Y sin actualizar

Lectura Sucia o Fantasma

T1:	T2:	T1	T2
read_item(X);		4	
X ← X - N;		2	
write_item(X);		2	
	read_item(X);		2
	X ← X- N;		0
	write_item(X);		0
Falla!!			

N=2

Al fallar T1, el valor X=2 virtualmente nunca existió !! → T2 leyó un valor "fantasma"

Plan de Ejecución de Transacciones

- Serializabilidad de Transacciones → el resultado de la ejecución concurrente de dos o más transacciones debe ser igual al que se hubiese obtenido de una ejecución serial (no concurrente)
- Si hay N transacciones ejecutándose concurrentemente → N! ejecuciones seriales posibles

Serializabilidad

- Idea: cuáles operaciones pueden intercambiarse en un plan de ejecución?
- Las siguientes NO, pues el resultado cambia !!:
  - Acciones de la misma transacción
  - Acciones en diferentes transacciones sobre el mismo ítem, cuando al menos una de las acciones es una escritura

Serializabilidad de los Planes de Ejecución

- Serial
  - Un plan es serial si para cada transacción participante, todas las operaciones se realizan consecutivamente en el plan (no hay intercalado de sentencias).
- Serializable
  - Un plan es serializable si es equivalente a cualquier plan serial de las mismas transacciones.

Bloqueos (locks)

- Cerrojo (Candado) que crea una zona de exclusión en determinados ítems de datos.
  - El tamaño del bloqueo define la granularidad: fina (nivel de datos) → → gruesa (BD)
- Cerrojo: variable asociada a cada ítem de dato
  - Describe el estado de un ítem con respecto a las operaciones que pueden realizarse sobre él
- Cerrojo binario: Cerrado/Abierto (Locked/Unlocked)
- Cerrojo multimodo: Cerrado para lectura/Cerrado para escritura/Abierto (ReadLock / WriteLock / Unlocked)
- Tres operaciones
  - read_lock(X) → cerrojo aplicado al item X para que una transaccion lo pueda leer
  - write_lock(X) → cerrojo aplicado al item X para que una transaccion lo pueda escribir
  - unlock(X) → item X liberado para que otra transacción lo use.
- Cada ítem de dato puede estar en cualquiera de los tres estados.

Bloqueos (locks)

Granularidad

- Afecta significativamente el nivel de concurrencia
- Afecta significativamente el desempeño del control de concurrencia
- Ejemplos de granularidad de ítems:
  - Un campo de un registro de una BD.
  - Un registro de la BD.
  - Un bloque de disco.
  - Un archivo completo.
  - Toda la BD…..
- Los locks se almacenan en una tabla del sistema.

Bloqueos (locks)

- Los bloqueos sobre ítems de datos pueden combinarse limitadamente
  - S-locks (compartidos/shared) para lecturas: slock(x) (≡ReadLock(x))
  - X-locks (exclusivos) para escrituras: xlock(x) (≡WriteLock(x))

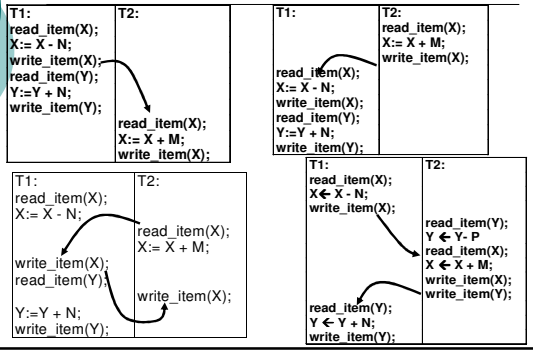
Compatibilidad de bloqueos

Bloqueo en el ítem \ bloqueo solicitado	-	S	X
-	Ok	Ok	Ok
S	Ok	Ok	---
X	Ok	---	---

Control de Concurrencia: Estrategias

- Los DBMSs no verifican la serializabilidad
  - Muy ineficiente o impracticable !! (estrategia reparadora) →
    - Las transacciones arriban continuamente
    - Debería 'ir probando' todas las combinaciones
  - Solución: *protocolos* (estrategia preventiva)
- Se puede garantizar la serializabilidad de los planes de ejecución si cada transacción sigue el protocolo y éste es forzado por el procesador de transacciones.

Grafos de Precedencia



Grafo de Serializabilidad

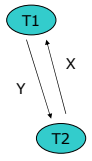
- Un nodo para cada transacción T_i
- Un arco de T_i a T_j si hay una operación de T_i que precede y entra en conflicto con una de T_j
- Teorema: Un plan de ejecución es serializable si y sólo si su grafo de serializabilidad es acíclico.

Locks no evitan todos los problemas …

Ejecutadas según un plan concurrente, con cerrojos

T1	T2
read_lock(Y);	
read_item(Y);	
unlock(Y);	
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	Y ←X+Y;
	write_item(Y);
	unlock(Y);
write_lock(X);	
read_item(X);	
X ←X+Y;	
write_item(X);	
unlock(X);	

Desbloquea demasiado pronto !!



Resultado: X=50, Y=50

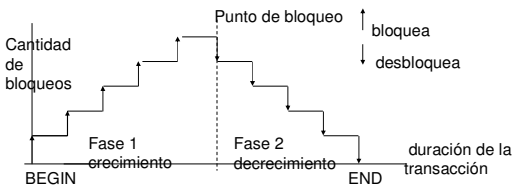
ciclo en el grafo → No-serializable!

Protocolo de Bloqueo en 2 Fases (P2F)

- Se dice que una transacción sigue el protocolo de dos fases si TODOS los bloqueos (fase de crecimiento) preceden al primer desbloqueo (fase de decrecimiento)

Protocolo de Bloqueo en 2 Fases (P2F)

- Básico: Cuando una transacción devuelve un ítem, ya no puede bloquear ningún otro



Ejemplo: Dos transacciones

T1	T2
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
unlock(Y);	unlock(X);
write_lock(X);	write_lock(Y);
read_item(X);	read_item(Y);
X ← X+Y;	Y ←X+Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y);

Ejecutadas según un plan serial S1: T1, T2
Valores iniciales X=20, Y=30 → Resultado: X=50, Y=80

Ejemplo: Dos transacciones según el P2F

T1'	T2'
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
write_lock(X);	write_lock(Y);
unlock(Y);	unlock(X);
read_item(X);	read_item(Y);
X ← X+Y;	Y ← X+Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y);

- T1' y T2' siguen el P2F
- Cualquier plan de ejecución de T1' y T2' será serializable
- La concurrencia es limitada por la necesidad de arbitrar requerimientos de ambas transacciones

Abrazo Mortal (deadlock)

- Dos o más transacciones esperan indefinidamente que otra devuelva un ítem de dato.

T1	T2
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
no puede → espera !!	
	write_lock(Y);
	no puede → espera !!

Abrazo Mortal (deadlock) e Inanición (livelock)

- Para prevenir deadlocks →
  - P2F conservativo ← obtener TODOS los recursos al iniciar la transacción
  - Timestamping → las transacciones más jóvenes son abortadas
- Detección de deadlocks →
  - Grafo espera-por
  - Selección de una 'víctima'
  - Recomienzo cíclico

- Livelock: una transacción no puede obtener los recursos que necesita por un período de tiempo indefinido, mientras las otras avanzan normalmente →
  - Estrategia → first-come-first-served