Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo

Bases de Datos

Pequeños apuntes de SEGURIDAD e INTEGRIDAD[*]

1 SEGURIDAD, INTEGRIDAD Y CONFIDENCIALIDAD

- En un SBGD hay datos que son accedidos por muchos usuarios
- Debe evitarse que haya problemas con los datos:

- **Seguridad**: Fallos lógicos o físicos que destruyan los datos.
- **Integridad**: Modificaciones que lleven a la BD quede a un estado inconsistente.
- **Confidencialidad**: Acceso no autorizado a la información.

1.1 CONFIDENCIALIDAD

- Evitar el acceso no autorizado (malintencionado):

- Lectura (robo de información)
- Modificación: inserción, borrado, ...

MEDIDAS DE SEGURIDAD

- **Físicas**: Controlar el acceso al equipo. Tarjetas de acceso, etc.
- **Personal**: Acceso sólo del personal autorizado. Evitar sobornos, etc.
- **SO**: Seguridad a nivel de SO
- **SGBD**: Uso herramientas de seguridad que proporcione el SGBD. Perfiles de usuario, vistas, restricciones de uso de vistas, etc.

VISTAS Y AUTORIZACIONES

- El SGBD debe distinguir diferentes usuarios.
- Las vistas permiten dar acceso parcial a una parte de la BD.
- Las autorizaciones (privilegios) indican restricciones a lo que cada usuario puede hacer con una vista.

[Ver capítulo de vistas en SQL para saber qué tipo de restricciones se pueden imponer a las vistas y cómo se especifican los privilegios: GRANT y REVOKE]

[*] En caso de encontrar alguna errata de cualquier tipo en estos apuntes (que seguro que la habrá), puede comunicarse al que los ha perpretado: Darío Álvarez, correo-e: darioa@uniovi.es

CIFRADO

- El cifrado (codificación) de la información.
- Permite que aunque se pueda acceder a la información (cifrada) (ej: mirando el contenido físico del disco, captando mensajes por líneas de comunicación) sólo los autorizados puedan descifrar su verdadero contenido.

Características de un buen mecanismo de cifrado:

- **Sencillo**: Debe ser sencillo cifrar y descifrar los datos.
- **Clave**: La seguridad del sistema no depende de que se conozca el algoritmo de cifrado, sino un parámetro del mismo (la clave de cifrado)
- **Seguro**: Debe ser muy difícil descubrir la clave a partir del estudio de una información cifrada.

CLAVES PÚBLICAS

- La seguridad de un sistema basado en una clave depende de la seguridad del mecanismo de distribución de la clave (ej: si comunico al destinatario de la información cifrada la clave por teléfono y alguien la oye la seguridad del mecanismo desaparece).

- **Elección de las claves:**

- Existe un algoritmo eficiente para saber si un número es primo o no
- No existe un algoritmo eficiente para descomponer un número en sus factores primos (se tardaría demasiado tiempo)

- Clave pública P1*P2 (P1, P2 primos suficientemente grandes)
- Clave privada: (P1, P2)

- Es imposible obtener en un tiempo razonable P1 y P2 a partir de P1*P2.

1.2 SEGURIDAD

- Evitar pérdidas de datos por fallos hardware o software (fallo disco, etc.). Normalmente suelen ser fallos de disco o pérdida de memoria RAM.
- Aparte del punto de vista de los SGBD, intervienen otros niveles (ej: discos replicados, etc.)

- A pesar de estos posibles fallos la base de datos debe quedar siempre en un estado consistente.

TRANSACCIONES

- Secuencia de operaciones que se ejecutan completamente o bien no se realizan.
- No puede quedarse en un estado intermedio.
- Ej: una transferencia entre dos cuentas no puede quedarse en un estado intermedio: O se deja el dinero en la primera cuenta o en la segunda, pero no se puede sacar el dinero de la primera cuenta, que falle algo en ese momento y no entregarlo en la segunda.

- Cuando una transacción finaliza con éxito, se graba (COMMIT)
- Si fracasa, se restaura el estado anterior (ROLLBACK)

Propiedades de una transacción:

- **Atomicidad**: Se realizan o todas las instrucciones o ninguna.
- **Consistencia** (Corrección, Preservación consistencia): La transacción siempre deja la BD en un estado consistente (si no lo hace puede ser por errores lógicos o físicos)

y además:

- **aIslamiento**: Los efectos de una transacción no se ven en el exterior hasta que esta finaliza.
- **Durabilidad** (Persistencia): Una vez finalizada la transacción los efectos perduran en la BD.
- **Seriabilidad**: La ejecución concurrente de varias transacciones debe generar el mismo resultado que la ejecución en serie de las mismas.

ej: Transacción 1 - Restar 10 Transacción 2 - Sumar 10

T1.1 Read(x);		T2.1 Read(x)
T1.2 x:= x-10;	x=10	T2.2 x:=x+10;
T1.3 Write(x);		T2.3 Write(x);

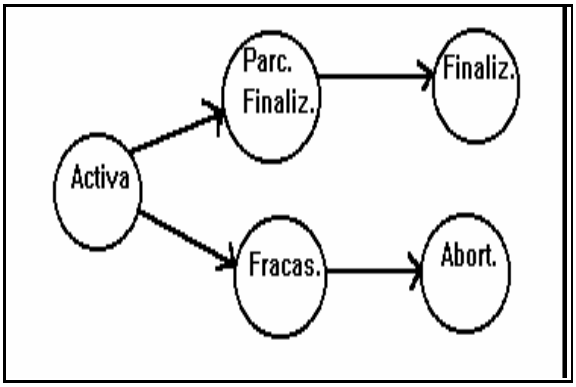
- El resultado final del dato común x debe ser 10 (el resultado de realizar T1 y luego T2 o viceversa).
- En una ejecución concurrente de transacciones no se puede permitir que el resultado sea distinto (Problema típico de sincronización de procesos, independencia en el tiempo, exclusión mutua, etc.)

- Para ello necesitamos mecanismos para, en el supuesto de que se necesite para no caer en el caso anterior:
  - Deshacer transacciones

- Rehacer transacciones

Estados de una transacción:

- **Activa**: Estado inicial (realizando sus instrucciones)
- **Parcialmente finalizada**: Tras acabar la última instrucción
- **Fracasada**: Tras descubrirse que no puede continuar la ejecución normal, por ejemplo por un error lógico (violación restricción de integridad).
- **Abortada**: Tras deshacer la transacción y devolver la BD al estado anterior (consistente).
- **Finalizada**: Tras completarse con éxito (cambios visibles).



- Una vez abortada hay dos opciones:
- Rehacer la transacción: Si fue abortada por un error HW o Software ajenos a la transacción.
- Eliminar la transacción: Si fue abortada por un error interno de la transacción (violación restricción de integridad, etc.).

ATOMICIDAD DE LAS TRANSACCIONES

- Existen diferentes mecanismos para asegurar la atomicidad de las transacciones.

FICHEROS DE DIARIO (*LOG*)

- En lugar de grabar (posiblemente) sólo en la base de datos las operaciones de una transacción, se graban también en un fichero de diario (histórico o *log*).
- El fichero de diario debe estar en almacenamiento estable (confiable, no se puede perder la información).

- Se almacenan registros que describen las modificaciones en la BD.
- Posteriormente se pueden usar esos registros para actualizar (o recuperar) el estado de la BD a través de los mismos.

Estructura (posible) de un registro de este tipo
- Nombre de la transacción
- Nombre del dato
- Valor antiguo
- Valor nuevo
- ...

Modificación retardada

- Se actualiza la BD después de que la transacción finaliza parcialmente (usando los datos del diario).
- Si falla algo durante la transacción (fallo del sistema, aborta la transacción) se omiten los registros correspondientes y la base de datos queda en el estado anterior.

Modificación inmediata

- Se actualiza directamente la BD.
- Si hay algún problema se deshacen las transacciones usando el *log* para devolverla al estado consistente anterior.

PÁGINAS COPIA (SOMBRA) (SHADOW PAGING)

- En sistemas que usan paginas de disco para la BD.
- Cuando una transacción modifica datos de una página de la BD, se crea una copia de la página y se modifica la original.
- Cuando la transacción finaliza se descartan las copias.
- Si hay que recuperar, se tienen los datos originales a través de la página copia.
- Puede usarse un esquema de modificación retardada a la inversa.
- Las páginas pueden ser páginas puras de disco o bien páginas del sistema de memoria virtual.

1.3 INTEGRIDAD

- Mantener la consistencia de la base de datos a nivel lógico:

- Operaciones semánticamente inconsistentes
  - Restricciones de integridad
- Problemas derivados de la concurrencia
- Seriabilidad de las transacciones concurrentes
- En el ejemplo anterior, un esquema de planificación como este: T1.1, T2.1, T1.2, T2.2, T1.3, T2.3 no obtiene el resultado correcto (10) y en su lugar obtiene 20.
- El resultado es diferente del de una ejecución serializada de T1 y T2.

- **Esquema serial**: Aquél que obtiene el resultado semánticamente correcto
- **Esquema no serial**: Aquél que no lo obtiene (por ejemplo el anterior)

TÉCNICAS DE CONTROL DE CONCURRENCIA

- Similares a las aplicadas en sistemas operativos.
- Se usan para asegurar la seriabilidad en la ejecución de transacciones concurrentes.

- Técnicas pesimistas: Evitan que se produzcan los problemas, como bloqueos (cerrojos)...
- Técnicas optimistas: Sólo actúan en caso de que aparezca algún problema.

Pueden estar totalmente controladas por el programador, o realizadas internamente por el SGBD al procesar las transacciones o de una forma mixta.

BLOQUEOS (CERROJOS, *LOCKS*)

- Indican para cada variable de datos su estado respecto a las posibles operaciones que se pueden realizar con ella en cada momento.

- Se trata de evitar el acceso a un dato por otras transacciones (cuando este acceso produzca problemas) bloqueando el dato al resto. Reteniendo un cerrojo o bloqueo sobre el dato.
- Es similar a las exclusiones mutuas.
- El SGBD lo puede hacer automáticamente en algunos casos.
- Puede ser establecido arbitrariamente en las transacciones.

Tipos de bloqueo

- **Exclusivo** (Escritura)

El dato no puede ser accedido por otra transacción, ni siquiera para adquirir un bloqueo.
Normalmente se usa cuando hay que actualizar un dato.
Una vez que se tiene un bloqueo exclusivo sobre un dato, este no puede ser usado por otra transacción.

- **Compartido** (Lectura)

Otras transacciones pueden consultar el dato, pero no modificarlo (adquiriendo previamente un bloqueo compartido sobre el dato).

Una vez que se tiene un bloqueo compartido sobre un dato, este no sólo puede ser consultado por otra transacción.

(Similar al problema de los redactores y lectores)

Bloqueo en dos fases

- Es un protocolo que usa bloqueos y asegura la seriabilidad.
- El uso de bloqueos únicamente no la garantiza. La transacción puede estar mal diseñada a nivel lógico, etc.

- Cada transacción puede hacer peticiones de bloqueos y desbloqueos en dos fases:

- **Fase de crecimiento**: Una transacción puede obtener un bloqueo, pero no liberar ninguno.
- **Fase de encogimiento**: Una transacción puede liberar bloqueos, pero no adquirirlos.

- Se trata de pedir todos los bloqueos antes de empezar a soltarlos.

- Esta técnica logra la serialización, pero no evita interbloqueos (ej: 2 transacciones piden bloqueos cruzados).
- Para evitar interbloqueos pueden usarse otras técnicas (SO), o bien codificar las transacciones cuidadosamente...

Granularidad: fina/gruesa

- Determinada por el tamaño del elemento de datos que se bloquea (comparte):
    - Campo de un registro (Atributo)
    - Registro (Tupla)
    - Conjunto de registros (Conj. de tuplas)
    - Fichero (Relación)
    - Toda la base de datos

- Cuanto más fina es la granularidad, mayor es el grado de concurrencia posible, pero se complica la gestión de los bloqueos al ser éstos más.
- Ej: Si lo que se bloquea es a nivel de toda la base de datos, el control es muy sencillo pero sólo puede trabajar una transacción a la vez con la base de datos, aunque sólo use una pequeña parte de ella.

TÉCNICAS OPTIMISTAS

- Se permite el acceso libre de las transacciones a los objetos.
- Se determina antes de finalizarlas si ha habido interferencias (problemas) y se finalizan unas y se relanzan otras de manera adecuada.

Fases de las transacciones en estos esquemas (normalmente):

- **Lectura**: Las transacciones hacen operaciones en copias privadas de los objetos.
- **Validación**: Se comprueba si el conjunto de los objetos tratados en una transacción se solapa con el de los modificados en otra (que ya los haya validado) (Condiciones de Bernstein)
- **Grabación**: Si no hay interferencias, se graban las modificaciones. Si las hay, se relanza la transacción.

- En lugar de evitar los problemas, sólo se actúa en el caso de que verdaderamente los haya.

- El uso de técnicas optimistas o pesimistas dependerá de la probabilidad de conflictos. Si esta es alta, con técnicas optimistas habrá que relanzar muchas transacciones, desperdiciando el trabajo anterior.

MARCAS DE TIEMPO (TIMESTAMPING)

- Es una técnica que podría encuadrarse dentro de las optimistas.
- Se asigna a cada transacción un identificador único, su marca de tiempo (tiempo de inicio).
- No hay bloqueos. Se retardan las actualizaciones hasta el final de la transacción.
- Si una transacción quiere actualizar o consultar un dato que ha sido actualizado por una transacción de fecha posterior (más reciente), entonces se deshace y se vuelve a lanzar.
- Es decir si se intenta usar un dato que se ha modificado después de que la transacción se inicie, no se puede continuar porque el valor inicial que tenía el dato para la transacción ha cambiado.

- Este mecanismo asegura la seriabilidad al ordenar las transacciones evitando solapamientos, basándose en el momento de inicio de cada transacción.
- Además, no se pueden producir interbloqueos al no usar mecanismos que impliquen la espera de las transacciones.