



Transacciones



¿Qué es una transacción?

Es una unidad lógica y atómica de procesamiento de la base que datos que puede incluir una o más operaciones.

Transacción T1

Begin

```
INSERT INTO T1 VALUES (1, 'Juan')  
DELETE FROM T2 WHERE id=1
```

End

Transacción T2

Begin

```
UPDATE T1 SET Precio=0 where id=1
```

End

Control de una transacción



El SGBD debe controlar las transacciones para no alterar la consistencia de la base de datos, ya que las transacciones puede utilizar los mismos recursos.

Saldo Inicial: 0 pesos

T1: Deposita Dinero

```
INSERT INTO MOVIMIENTO  
(NroCuenta, TipoMov, Monto)  
VALUES (1, 'D', 10000)
```

```
UPDATE SALDO  
SET MSaldo=MSaldo + 10000  
Where NroCuenta=1
```

T2: Extrae Dinero

```
INSERT INTO MOVIMIENTO  
(NroCuenta, TipoMov, Monto)  
VALUES (1, 'E', 10000)
```

```
UPDATE SALDO  
SET MSaldo=MSaldo - 10000  
Where NroCuenta=1
```

T3: Consulta el Dinero

```
Select MSaldo  
From SALDO  
Where NroCuenta=1
```

Iniciar una transacción




Transacción T1

BEGIN TRANSACTION

INSERT INTO T1 VALUES (1, 'Juan')

DELETE FROM T2 WHERE id=1

COMMIT TRANSACTION




**Finalizó sin errores y
se confirmó**

Transacción T2


BEGIN TRANSACTION

INSERT INTO T1 VALUES (1, 'Ana')



DELETE FROM T2 WHERE id=1

ROLLBACK TRANSACTION



**Finalizó con errores y
se deshizo**

Propiedades de una transacción



ACID (Atomicidad – Consistencia – Aislamiento – Durabilidad)

Atomicidad: una transacción es una unidad atómica de procesamiento, se realiza por completo (se confirma) o bien no se realiza en absoluto (se aborta).

Conservación de consistencia: una transacción conserva la consistencia si su ejecución completa lleva la base de datos de un estado consistente a otro. Una transacción no puede violar la integridad de la base de datos. Por ejemplo: no puede dejar el Stock en negativo.

Aislamiento (en ingles es **Isolation**): una transacción debería parecer que se está ejecutando en forma aislada de las demás transacciones. Es decir, la ejecución de una transacción no debe interferir con la ejecución de otra transacción, debe dar el mismo resultado que si se ejecuta en serie (una después de la otra). Ningún cambio será revelado al resto de las transacciones hasta que la transacción haya sido finalizado correctamente.

Durabilidad: los cambios aplicados en la base de datos por una transacción confirmada deben perdurar en la base de datos. Estos cambios no deben perderse por un fallo posterior.



Funciones de del SGBD

El SGBD es responsable de realizar:

- **Control de Concurrencia (Transaction Manager)**
- **Control de recuperación (Recovery Manager)**

Problemas de la concurrencia



Problema de la actualización Perdida (Lost Update):

T1	T2
Leer (X)	
$X = X - N$	
	Leer (X)
	$X = X + M$
Escribir (X)	
Commit;	
	Escribir (X)
	Commit;

$X=10$
 $N=1$
 $M=2$

El cambio que hizo T1 en X se pierde porque T2 “pisa” el valor X



T2 no debería haber podido leer el valor de X si T1 lo estaba modificando

Problemas de la concurrencia



Problema de la actualización sucia (Dirty Reads):

T1	T2
Leer (R1,x)	
X=x-N	
Escribir (R1,x)	
	Leer (R1,x)
Rollback;	
	X=x+M
	Escribir (R1,x)

X=10
N=1
M=2



T2 no debería haber leído el valor de X si T1 aún no había confirmado su valor

Problemas de la concurrencia



Problema del resumen incorrecto:

T1 (suma saldos)	T2 (transferencia)
Leer(X)	
	Leer(X)
	x=x-1000
	Escribir(X)
	Leer(Y)
	Y = Y + 1000
	Escribir(Y)
	Commit;
Leer(Y)	
Leer(Z)	
sum=X+Y+Z	
Commit;	

X=1000
Y=2000
Z=3000



T1 tomó un valor en X anterior a la actualización y un valor X posterior.

Conflictos



Se dice que existe un conflicto entre dos transacciones, cuándo utilizan el mismo recurso y al menos, alguna de las dos quiere escribir el elemento.

- *T1: Leer(R1,x) / T2: Escribir(R1,x)*
- *T1: Escribir(R1,x) / T2: Leer(R1,x)*
- *T1: Escribir(R1,x) / T2: Escribir(R1,x)*



Lockeos (Bloqueos)

Es un mecanismo que permite que las transacciones puedan ejecutar en forma serializable, aunque no lo sean, basado en controlar el uso de recursos para que las transacciones que necesitan el mismo elemento puedan aguardar a obtenerlo si otra transacción lo tiene en uso.

Los principios del lockeo son:

- *Obtener un lockeo antes de acceder al recurso*
- *Luego de usarlo, liberar el recurso*



Compatibilidad de Lockeos

Tipos de Lockeos:

- *Lock-S: Lockeo Shared o de Sólo Lectura*
- *Lock-X: Lockeo Exclusive ó de Escritura*

Reglas:

- *Si una transacción necesita leer un objeto, primero tendrá que solicitar un lockeo de Shared.*
- *Si una transacción necesita modificar un objeto, primero tendrá que solicitar un lockeo de X exclusive.*
- *Sólo se podrá tener un solo lockeo X por objeto, pero múltiples S por objeto.*

	Shared	Exclusive
Shared	S	N
Exclusive	N	N



Lockeos en SGBD

Los SGBD realizan automáticamente los tipos de lockeos. Generalmente, estos son los tipos de lockeos que establecen según la operación realizada y según el nivel de aislamiento establecido:

- *select → no realiza lockeos (*)*
- *insert/update/delete → xlock /row exclusive*
- *commit / rollback → libera todos los lockeos*

(*) Dependiendo del nivel de aislamiento



¿Siempre se establecen este tipo de lockeos? ¿De qué depende?



Niveles de Aislamiento

Pueden ocurrir 3 tipos de violaciones en los datos:

- *Lectura sucia (dirty reads): es cuando una transacción lee un dato, luego otra lee el dato actualizado, pero la primera aborta, por lo cual la segunda transacción leyó aun dato que no fue aplicado a la base.*
- *Lectura no repetible (nonrepeatable read): es cuando una transacción lee un valor, luego otra transacción actualiza ese valor y si la primer transacción volviera a leer ese mismo item, ya no tendría el mismo valor original.*
- *Lectura fantasma (phantom reads): es cuando una transacción lee un conjunto de filas de una table que cumplen una determina condicion en la cláusula where. Luego, otra transacción inserta nuevas filas en esa table las cuales tambien cumple esa misma condición, por lo cual si la primer transaccion volviera a querer leer las filas que cumplen la condicion, le van aparecer filas nuevas (filas fantasmas) que antes no estaban.*



Niveles de Aislamiento

SET TRANSACTION ISOLATION LEVEL <Nivel de Aislamiento>

	Dirty Reads	NonRepeatable Read	Phantom Reads
Read Uncommitted	SI	SI	SI
Read Committed	NO	SI	SI
Repeatable Read	NO	NO	SI
Serializable	NO	NO	NO



Lockeos: Granularidad

Además del tipo de lockeo que podrá ser Shared(S) ó Exclusive(X), el SGBD establecerá qué debe lockear, lo que llamaremos la granularidad del lockeo:

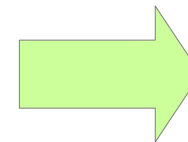
- *Database - Base de datos*
- *Table – Tabla*
- *Page - Página/s ó Bloque/s*
- *Row – Fila*



Lockeos: Deadlock

Se produce cuando cada transacción T en un conjunto de dos o más transacciones está esperando a algún elemento que está bloqueado por alguna otra transacción T' de dicho conjunto.

T1	T2
Lock-X(X)	
Read(X)	
Write(X)	
	Lock-X(Y)
	Read(Y)
	Write(Y)
Lock-X(Y)	
Read(Y)	
Write(Y)	
	Lock-X(X)
	Write(X)



DEADLOCK



Lockeos: Deadlock

¿Qué hacer con los deadlocks?

- **Ignorarlos:** Dejar que los programadores lo manejen.
- **Prevenirlo:** Garantizar que los deadlocks jamás puedan ocurrir y esto se podría realizar lockeando todos los recursos antes de usarlo, también se podría realizar los lockeos en un orden estricto.
- **Evitarse:** Detectar los potenciales deadlocks en forma adelantada y tomar una acción sobre las transacciones.
- **Detectar y recuperar:** permitir los deadlocks, detener y recuperar las transacciones.



Control de recuperación

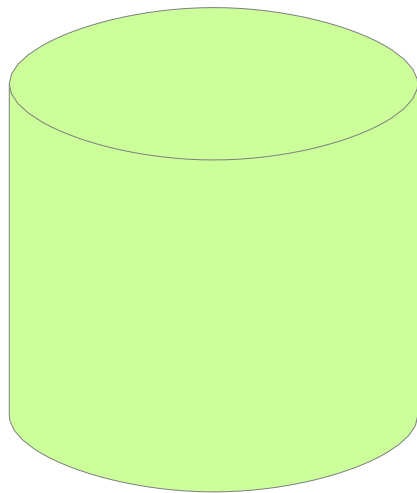
Dado que esas transacciones que están ejecutándose pueden cancelarse por un factor intencional o un factor externo, el SGBD debe garantizar la consistencia de los datos que estén siendo utilizados en el momento de la interrupción. Para ello posee un módulo de Recovery Manager encargado de dicha gestión.



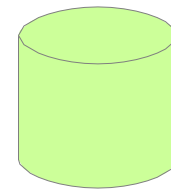
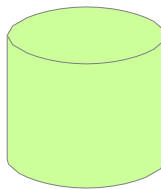
Log de Transacciones

Los SGBD poseen una bitácora de todas las operaciones que se van realizando en el motor. Esta bitácora es uno o más archivos físicos llamados archivos de LOG en donde guardan las operaciones.

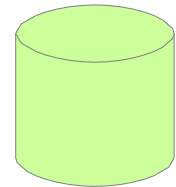
Cada vez que se inicia una transacción que realiza cambios, dichos cambios se van almacenando en el archivo de LOG.



Datafiles



...



Logfiles



Log de Transacciones

¿Qué se almacena?

Todas las operaciones que modifican datos (INSERT, UPDATE, DELETE, etc) hayan sido o no confirmadas con COMMIT.



Ante una falla

Cuando se produce una falla y el SGBD se reinicia, es necesario verificar las transacciones para reconstruir el estado al momento de la falla, sin generar inconsistencia de datos. Toda esta información se obtiene verificando el Log de Transacciones:

- La transacción T_i tiene que deshacer todo lo que había realizado, si la transacción había sido abortada $\langle T_i \text{ rollback} \rangle$
- La transacción T_i tiene que aplicar todos los cambios realizados si la transacción había sido commiteada y aún no se habían escrito los bloques cambiados $\langle T_i \text{ commit} \rangle$
- La transacción T_i tiene que deshacer todo lo realizado si no se ha encontrado un fin de esa transacción, ya sea por commit o rollback.

Checkpoint



Cuando cambiamos los bloques de la base de datos, esos bloques no se envían a disco, hasta que no se produce un evento de CheckPoint. El checkpoint es el evento por el cual los bloques modificados son realmente llevados a disco.

En los SGBD se puede establecer el tiempo en el cual el checkpoint se provoca. Cuanto más bloques tenga sin actualizar en disco, mayor será el tiempo de recuperación de la base de datos, en el caso de una falla.

Cuando el CheckPoint se produce, se podrá liberar esa sección del log que ya no se necesitará porque los bloques ya están actualizados.

Proceso de Recovery

