Manejo de transacciones

Unidad II.

Concepto de transacción

- Una transacción es un conjunto de operaciones que se ejecutan en una base de datos, cuyo resultado final debe ser que se ejecuten **todas** o **ninguna** de ellas.

Concepto de transacción

- No todas las operaciones SQL son transaccionales.

Solo son transaccionales las siguientes operaciones :

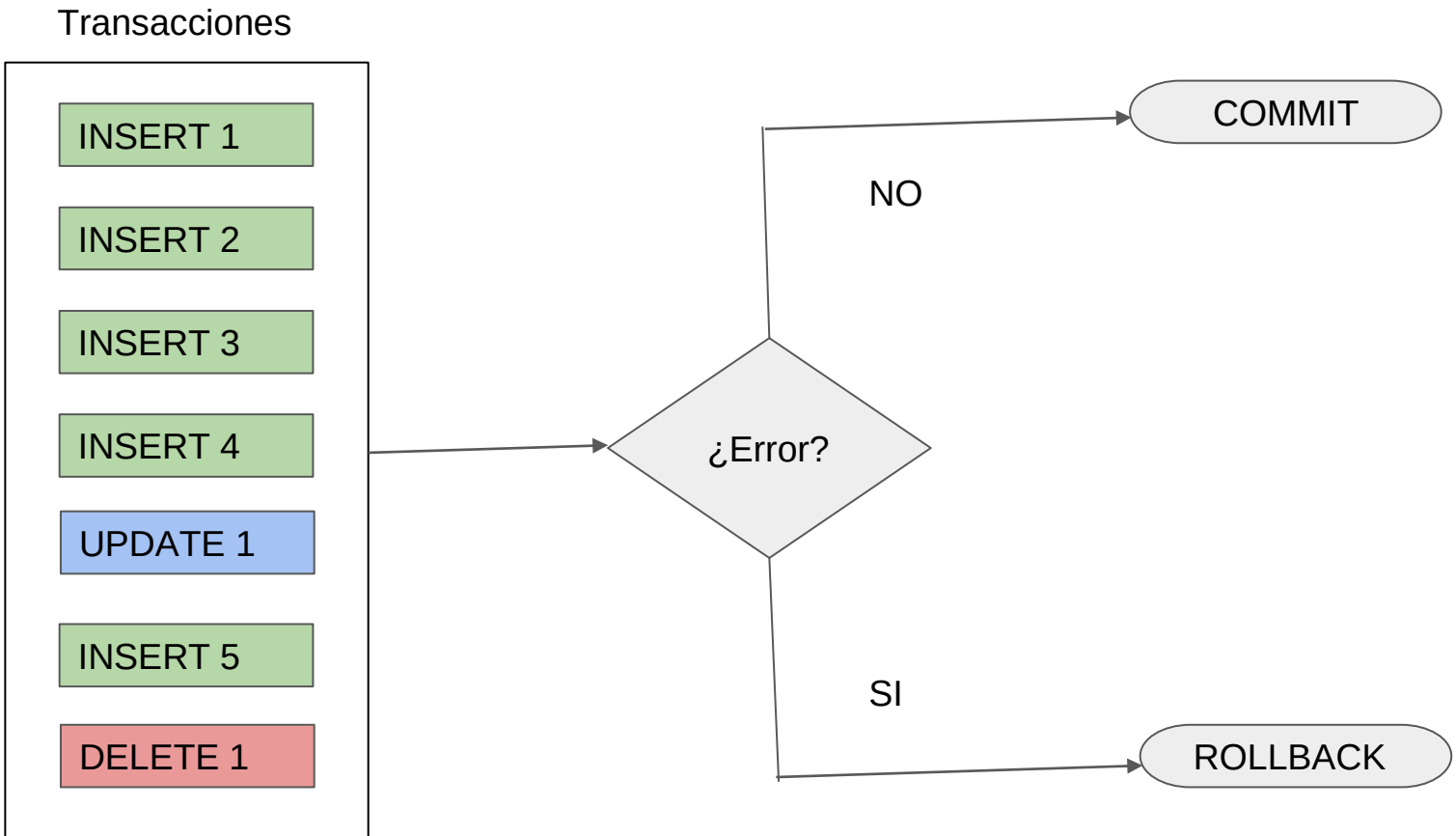
✓ Select
✓ Insert
✓ Update
✓ Delete

Commit y rollback

- Para confirmar una transacción se utiliza la sentencia **COMMIT.** Cuando realizamos **COMMIT** los cambios se escriben en la base de datos.

- Para deshacer una transacción se utiliza la sentencia **ROLLBACK.** Cuando realizamos **ROLLBACK** se deshacen todas las modificaciones realizadas por la transacción en la base de datos, quedando la base de datos en el mismo estado que antes de iniciarse la transacción.

Commit y rollback



Ejemplo: Transferencias bancarias.

Para realizar una transferencia de dinero entre dos cuentas bancarias se debe descontar el dinero de una cuenta, realizar el ingreso en la otra cuenta y grabar las operaciones y movimientos necesarios, actualizar los saldos.

Ejemplo: Bancario.

Disminuir el saldo de la cuenta de ahorro

```
UPDATE savings_accounts
SET balance = balance - 500
WHERE account = 3209;
```

Aumentar el saldo de cuenta

```
UPDATE checking_accounts
SET balance = balance + 500
WHERE account = 3208;
```

Registrar la transacción

```
INSERT INTO journal VALUES
(journal_seq.NEXTVAL, '1B'
3209, 3208, 500);
```

Transacción

Bancario. ***COMMIT***

Ejemplo

Disminuir el saldo de la cuenta de ahorro

if oK!!! then

Aumentar el saldo de cuenta

if Ok!!! then

Registrar la transacción

if Ok!!! then
COMMIT

Confirma

Si se pueden realizar las tres declaraciones SQL, Oracle,puede mantener las cuentas en un equilibrio adecuado, los efectos de la transacción pueden confirmarse o aplicarse a las tablas de la base de datos.

Bancario. ***ROLLBACK***

Ejemplo

Disminuir el saldo de la cuenta de ahorro

if oK!!! then

Aumentar el saldo de cuenta

if no Ok!!! then

ROLLBACK

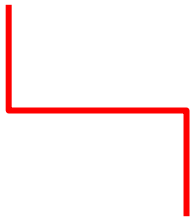
Registrar la transacción

Sin embargo, si un problema, tales fondos insuficientes, un número de cuenta no válido o una falla de hardware impide que se completen uno o dos de los extractos de la transacción, la transacción completa debe revertirse (anularse) para que el saldo de todas las cuentas sea correcto

Declaración Transacciones

- **COMMIT:** Se utiliza para hacer cambios permanentes en una base de datos.

```
BEGIN
  INSERT INTO pairtable VALUES (1, 2);
  COMMIT;
END;
```



Todas las funciones de la base de datos durante esa transacción son permanentes.

- **ROLLBACK:** Se utiliza para desechar cualquier cambio que se haya realizado en la base de datos después de COMMIT.
- Si la transacción falla, o termina con un ROLLBACK, entonces ninguna de las afirmaciones surte efecto.

```
BEGIN
  INSERT INTO pairtable VALUES (3, 4);
  ROLLBACK;
  INSERT INTO pairtable VALUES (5, 6);
  COMMIT;
END;
```

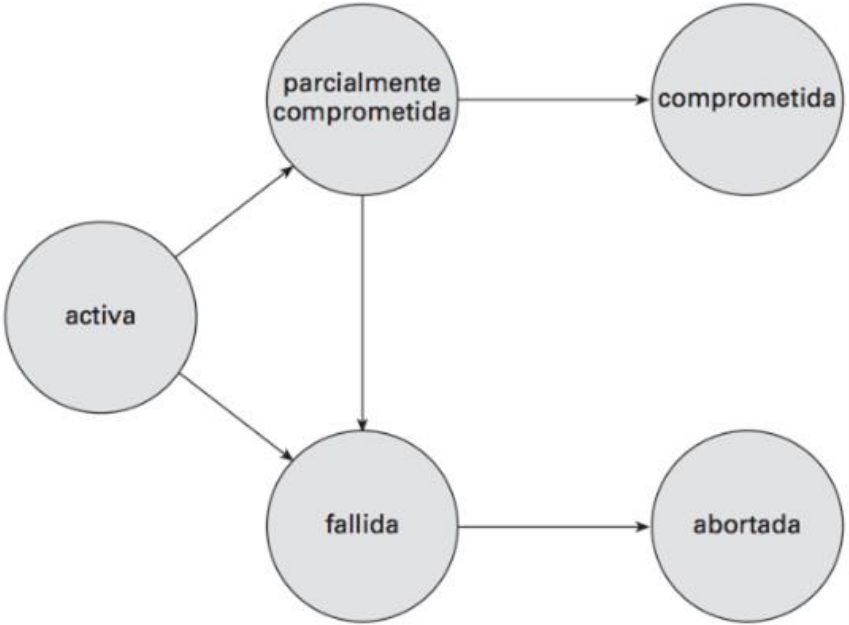
Declaración Transacciones

- **SAVEPOINT:** Se usa para marcar un punto intermedio en el procesamiento de transacciones.
- **ROLLBACK** se utiliza para devolver los valores de los datos al punto, de un punto de **SAVEPOINT.**


```
BEGIN
   INSERT INTO pairtable VALUES (7, 8);
   SAVEPOINT my_sp_1;
   INSERT INTO pairtable VALUES (9, 10);
   SAVEPOINT my_sp_2;
   INSERT INTO pairtable VALUES (11, 12);
   ROLLBACK to my_sp_1;
   INSERT INTO pairtable VALUES (13, 14);
   COMMIT;
END;
```

Estado de una transacción

- **Activa:** Estado inicial.
- **Parcialmente comprometida:** Después de ejecutar la última instrucción.
- **Fallida:** Al no poder continuar
- **Abortada:** Después de haber realizado rollback. BD restaurada.
- **Comprometida:** Completada con éxito.



Estado de una transacción

Al finalizar con una transacción abortada se tienen dos opciones:

- **Reiniciar,** sólo si se ha abortado por un error de hardware o software.
- **Cancelar,** si se encuentra un error lógico interno que sólo se corrige reescribiendo código.

Nota importante

ORACLE es completamente transaccional.

Esto es debido a que el funcionamiento de ORACLE se basa en el versionado de filas, por lo cual, al iniciar una transacción puede pasar todo el tiempo que sea necesario, que otras transacciones podrán realizar lecturas correctamente (accediendo a la versión correcta de cada filas).

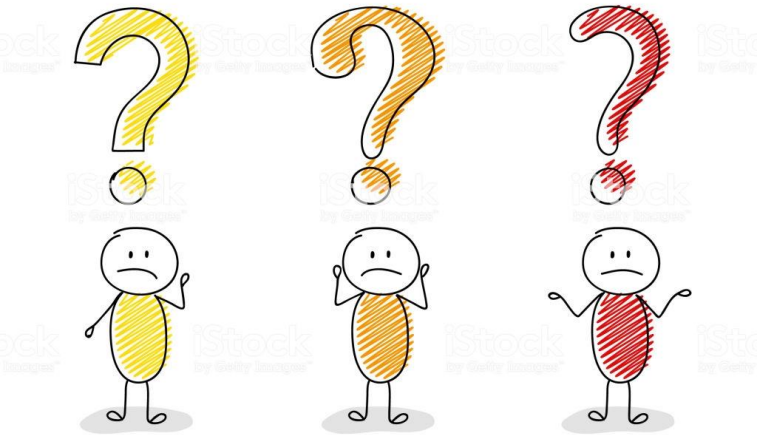
Nota importante

También es importante saber que, si alguna de las tablas afectadas por la transacción tiene triggers, las operaciones que realiza el trigger están dentro del ámbito de la transacción, y son confirmadas o deshechas conjuntamente con la transacción.

Durante la ejecución de una transacción, una segunda transacción no podrá ver los cambios realizados por la primera transacción hasta que esta se confirme.

Propiedades de transacción

- Cada transacción debe tener las propiedades ACID.
- ¿Cuáles son las propiedades ACID?



Propiedades de transacción

	Descripción
Atomicidad	La transacción es realizada completamente o no realiza ninguna acción. No puede realizar solamente mitad o parte de la transacción.
Consistencia	Solo son ejecutadas aquellas transacciones que no tiene conflicto con las reglas y directrices de integridad de la base de datos.
Aislamiento	Si hay dos o más transacciones que en un mismo tiempo deseen realizar cambio a una misma información, el sistema garantiza que cada transacción ignora al resto de las transacciones es decir que cada una se maneja de forma independiente para no generar errores.
Durabilidad	Cuando una transacción es exitosa los cambios hechos por la transacción permanecen en el sistema y no se pueden deshacer aunque falle el sistema.

Ejemplos de ACID

1. **Las transacciones bancarias que se estén realizando en el sistema serán visibles a todos los usuarios hasta que estas hayan sido declaradas finales, en la transacción bancaria es posible que el sistema este programado para intentar en 5 o 10 ocasiones antes de de abortar una transacción por completo.**

1. **Se transfieren fondos de una cuenta a otra, la transacción puede fallar por múltiples motivos, pero no deben restarse los fondos de una cuenta si no se ha sumado a la otra y al revés por lo tanto o se culmina la transacción o se niega la operación.**

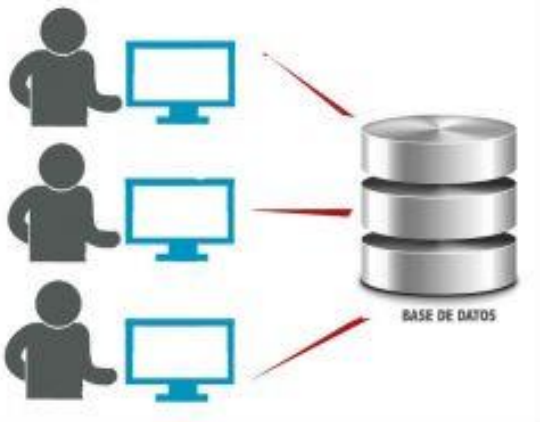
Ejemplos de ACID

3. **Al cumplir la operación bancaria y ocurre una falla del sistema como consecuencia, el sistema es capaz de acceder a dicha operación sin perder ninguna información.**

3. **Cada vez que se realice una transferencia bancaria será necesario notificar a la sucursal para actualizar la información para que lal transferencia sea exitosa, si no es posible comunicarse y actualizar la información en la sucursal del cliente, toda la transacción será abortada.**

¿Qué es concurrencia?

Es cuando muchas transacciones acceden a la misma base de datos al mismo tiempo. Especialmente, cuando acceden a los mismos datos.



Concurrencia

- Los sistemas de procesamiento de transacciones permiten normalmente la ejecución de varias transacciones concurrentemente.

- Permitir varias transacciones que actualizan concurrentemente los datos pueden provocar complicaciones en la consistencia de los mismos.

Tipos de lecturas

Cuando dos transacciones distintas intentan acceder concurrentemente a los mismos datos pueden ocurrir los siguientes problemas:

Lectura sucia
Lectura no repetible
Lectura fantasma

Tipos de lecturas

Lectura sucia (Dirty Read): Sucede cuando una segunda transacción lee datos que están siendo modificados por una transacción antes de que haga COMMIT.

Transacción 1	Transacción 2
`UPDATE cuentas SET saldo = saldo - 100 WHERE id = 1;`	
	`SELECT saldo FROM cuentas WHERE id = 1;`
`ROLLBACK`	

Tipos de lecturas

Lectura no repetible (Non-Repeatable Read): Se produce cuando una transacción consulta el mismo dato dos veces durante la ejecución de la transacción y la segunda vez encuentra que el valor del dato ha sido modificado por otra transacción.

Transacción 1	Transacción 2
`SELECT saldo FROM cuentas WHERE id = 1;`	
	`UPDATE cuentas SET saldo = saldo - 100 WHERE id = 1;`
`SELECT saldo FROM cuentas WHERE id = 1;`	

Tipos de lecturas

Lectura fantasma (Phantom Read): Este error ocurre cuando una transacción ejecuta dos veces una consulta que devuelve un conjunto de filas y en la segunda ejecución de la consulta aparecen nuevas filas en el conjunto que no existían cuando se inició lo transacción.

Transacción 1	Transacción 2
`SELECT SUM(saldo) FROM cuentas;`	
	`INSERT INTO cuentas VALUES (4, 3000);`
`SELECT SUM(saldo) FROM cuentas;`	

Ejemplos de tipos de lecturas

Se tiene en la tabla “productos” una columna denominada Stock cuyo valor inicial es 12 unidades, supongamos que dos transacciones concurrentes actualizan el valor del stock de la siguiente forma:

Transacción 1: compra de 100 unidades de ese recambio.
Transacción 2: venta de 5 unidades del recambio.

Ejemplo de lectura sucia

La ejecución normal de las transacciones debería ser:

TRANSACCION	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 1	Stock=12 + 100	112
Transacción 2	Leer Stock	112
Transacción 2	Stock=112-5	107

Cuando una transacción accede a un valor antes de que la transacción anterior haya finalizado:

Actualización perdida

TRANSACCIÓN	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 2	Leer Stock	12
Transacción 1	Stock =12 +100	112
Transacción 2	Stock=12-5	7

Ejemplo de lectura no repetible

Con los datos iniciales supongamos que la primera transacción se deshace. La ejecución correcta debería ser:

TRANSACCIÓN	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 1	Stock =12 +100	112
Transacción 1	ROLLBACK	12
Transacción 2	Leer Stock	12
Transacción 2	Stock =12 -5	7

Cuando una transacción se deshace después de que otra haya accedido a los datos:

Datos no comprometidos

TRANSACCIÓN	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 1	Stock =12 +100	112
Transacción 2	Leer Stock	112
Transacción 2	Stock =112 -5	107
Transacción 1	ROLLBACK	12

Ejemplo de lectura fantasma

Por ejemplo, cuando una transacción incluye operaciones de agregado (sumar, contar, etc.) sobre unos datos, mientras otra transacción los actualiza. Si la función lee unos datos antes de actualizar y otros después el resultado es inconsistente.

Transacción 1: Calcula la cantidad total disponible de Stock.
Transacción 2: Actualiza en 30 unidades la cantidad disponible de Stock de dos de esos productos: Z y V.

Ejemplo de lectura fantasma

La ejecución normal de estas transacciones es:

IdRecambio	VALOR Stock	VALOR AGREGADO
X	12	12
Y	15	15
Z	20	20+30=50
V	35	35-30=5
W	50	50
U	40	40
TOTAL	**172**	**172**

Ejemplo de lectura fantasma

Cuando la transacción lee unos datos antes de ser modificados y otros después:

	TRANSACCIÓN	OPERACIÓN	VALOR Stock	VALOR AGREGADO
	Transacción 1	Leer Stock para el IdRecambio X	12	12
	Transacción 1	Leer Stock para el IdRecambio Y	15	27
Lectura posterior a la modificación	Transacción 2	Leer Stock para el IdRecambio Z	20	
	Transacción 2	Modifica el Stock =20+30		
	Transacción 1	Leer Stock para el IdRecambio Z	50	77 (X+Y+Z después de modificar)
Lectura anterior a la modificación	Transacción 1	Leer Stock para el IdRecambio V	35	112 (X+Y+Z+V antes de modificar)
	Transacción 2	Leer Stock para el IdRecambio V	35	
	Transacción 2	Modifica el Stock=35-30	5	
	Transacción 1	Leer Stock para el IdRecambio W	50	162
	Transacción 1	Leer Stock para el IdRecambio U	40	**202**

Como vemos, el resultado final no es correcto porque el valor agregado excede en 30 unidades el resultado calculado anteriormente.

Niveles de aislamiento

Para evitar que sucedan los problemas de acceso concurrente se pueden establecer diferentes niveles de aislamiento que controlan el nivel de bloqueo durante el acceso a los datos. El estándar ANSI/ISO de SQL (SQL92) define cuatro niveles de aislamiento.

- **LECTURA SIN CONFIRMAR (READ UNCOMMITTED),**
- **LECTURA CONFIRMADA (READ COMMITTED),**
- **LECTURA REPETIBLE (REPEATABLE READ) o**
- **SERIALIZABLE.**

Niveles de aislamiento

LECTURA SIN CONFIRMAR: Apenas transaccional, permite hacer lecturas sucias (dirty reads), donde las consultas dentro de una transacción son afectadas por cambios no confirmados (not commited) de otras transacciones.

Niveles de aislamiento

LECTURA CONFIRMADA: Sólo se permiten lecturas de datos comprometidos.

Menos restrictivo. Cada sesión ve los cambios de las otras cuando éstas han hecho commit.

Es el valor por defecto.

Niveles de aislamiento

LECTURA REPETIBLE: Dentro de una transacción todas las lecturas son consistentes.

Niveles de aislamiento

SERIALIZABLE: Mayor nivel de aislamiento. Las transacciones se aíslan completamente.

Los cambios realizados en otros terminales no afectan, aunque las otras sesiones hayan hecho commit y por tanto hayan grabado físicamente los cambios.

Si se intenta modificar una fila ya modificada por otra sesión se obtiene un error "No se puede serializar el acceso para esa transacción".

Niveles de aislamiento

La siguiente tabla muestra los problemas de lectura que pueden ocurrir en cada uno de los modos de aislamiento.

Nivel	Sucia	No Repetible	Fantasma
`serializable`	NO	NO	NO
`repeatable read`	NO	NO	SI
`read commited`	NO	SI	SI
`read uncommited`	SI	SI	SI