Tema 6.

CONCEPTOS DE PROCESAMIENTO DE TRANSACCIONES

TRANSACCIONES

Una **transacción** es una **unidad lógica de trabajo** o procesamiento (ejecución de un programa que incluye operaciones de acceso a la base de datos).

Una transacción es una secuencia de operaciones que llevan la base de datos desde un estado de consistencia[1] a otro estado de consistencia, por esto suele decirse también que la transacción es una **unidad lógica de integridad**.

Cuando múltiples transacciones son introducidas en el sistema por varios usuarios, es necesario evitar que interfieran entre ellas de forma tal que provoquen que la BD quede en un estado no consistente; desde este punto de vista, podemos ver una transacción como una **unidad lógica de concurrencia**.

Cuando ocurre un fallo que provoca la caída del sistema, en el momento en el que había varias transacciones en curso de ejecución, muy probablemente dejará erróneos los datos en la BD (estado inconsistente); en estas circunstancias, se debe garantizar que la BD pueda ser recuperada a un estado en el cual su contenido sea consistente, por esto una transacción es considerada también una **unidad lógica de recuperación**.

La idea clave es que una transacción debe ser **atómica**, es decir, las operaciones que la componen deben ser ejecutadas en su totalidad o no ser ejecutadas en absoluto.

Una sentencia de definición o manipulación de datos ejecutada de forma interactiva (por ejemplo utilizar el *SQL*Plus* de Oracle para realizar una consulta) puede suponer el inicio de una transacción. Asimismo, la ejecución de una sentencia SQL por parte de un programa que no tiene ya una transacción en progreso, supone la iniciación de una transacción.

Toda transacción finaliza con una operación de *commit* (confirmar) o bien con una operación de *rollback* (anular, abortar o revertir).
Tanto una operación como la otra puede ser de tipo explícito (si la propia transacción (su código) contiene una sentencia COMMIT o ROLLBACK) o implícito (si dicha operación es realizada por el sistema de forma automática, por ejemplo tras detectar una terminación normal (éxito) o anormal (fallo) de la transacción).

Por defecto, una vez finalizada una transacción, si todas sus operaciones se han realizado con éxito, se realiza un COMMIT implícito de dicha transacción; y si alguna de ellas tuvo problemas, se lleva a cabo un ROLLBACK implícito de la transacción (es decir, se deshacen todas las operaciones que había realizado hasta el momento del fallo).

[1] El conjunto de valores almacenados en la base de datos cumple toda restricción de integridad especificada en el esquema, así como cualesquiera otras restricciones que deban cumplirse en la base de datos.

PROPIEDADES (deseables) DE UNA TRANSACCIÓN

Se suele hacer referencia a estas como las propiedades ACID (por sus iniciales en inglés).

1. **Atomicidad**

   Todas las operaciones de la transacción son ejecutadas por completo, o no se ejecuta ninguna de ellas (si se ejecuta la transacción, se hace hasta el final).

2. **Consistencia**

   Una transacción T transforma un estado consistente de la base de datos en otro estado consistente, aunque T no tiene por qué preservar la consistencia en todos los puntos intermedios de su ejecución. Un ejemplo es el de la transferencia de una cantidad de dinero entre dos cuentas bancarias.

3. **Aislamiento** (**I**solation)

   Una transacción está aislada del resto de transacciones.

   Aunque existan muchas transacciones ejecutándose a la vez, cualquier modificación de datos que realice T está oculta para el resto de transacciones hasta que T sea confirmada (realiza COMMIT).

   Es decir, para cualesquiera T1 y T2, se cumple que

   - T1 ve las actualizaciones de T2 después de que T2 realice COMMIT, o bien
   - T2 ve las modificaciones de T1, después de que T1 haga un COMMIT

   Pero nunca se cumplen ambas cosas al mismo tiempo.

   Nota: esta propiedad puede no imponerse de forma estricta[2]; de hecho, suelen definirse *niveles de aislamiento* de las transacciones.

4. **Durabilidad**

   Una vez que se confirma una transacción, sus actualizaciones sobreviven cualquier fallo del sistema. Las modificaciones ya no se pierden, aunque el sistema falle justo después de realizar dicha confirmación.

El Subsistema de Recuperación del SGBD es el encargado de conseguir el cumplimiento de las propiedades de atomicidad y durabilidad de las transacciones.

La conservación de la consistencia es una propiedad cuyo cumplimiento han de asegurar, por un lado los programadores de base de datos, y por otro el Subsistema de Integridad del SGBD.

El Subsistema de Control de Concurrencia es el encargado de conseguir el aislamiento de las transacciones.

[2] Aunque lo estudiaremos en el tema de *Recuperación de Caídas en Sistemas de Base de Datos*, si se impone de forma estricta el *aislamiento* de las transacciones, se resuelve el *problema de la actualización temporal* y no es necesario realizar *reversiones en cascada*.

OPERACIONES DE UNA TRANSACCIÓN

Para hacer posible el control de la concurrencia de las transacciones, así como la recuperación del sistema tras fallos o caídas del mismo, es necesario almacenar cuándo se **inicia**, **termina**, se **confirma** o se **aborta** cada transacción, y además qué **modificaciones** de qué elementos de BD realiza cada una.

- INICIO DE TRANSACCIÓN

  Operación que marca el momento en el que una transacción comienza a ejecutarse.

- LEER o ESCRIBIR

  Operaciones de lectura/escritura de elementos de la base de datos, que se realizan como parte de una transacción.

- FIN DE TRANSACCIÓN

  Las operaciones de LEER y ESCRIBIR han terminado. En este punto se verifica si la transacción debe abortarse por alguna razón (si viola el control de concurrencia, por ejemplo), o bien si los cambios realizados por la transacción (hasta ahora en *buffers* de memoria volátil) pueden aplicarse permanentemente a la base de datos en disco (es decir, si puede realizarse una operación de CONFIRMAR (COMMIT)).

- CONFIRMAR

  La transacción terminó con éxito, todos los cambios que ha realizado se pueden confirmar sin peligro en la BD y ya no serán cancelados.

- ABORTAR

  La transacción terminó sin éxito y toda actualización que ha realizado se debe cancelar.

Algunas técnicas de recuperación necesitan operaciones adicionales como las siguientes:

- DESHACER

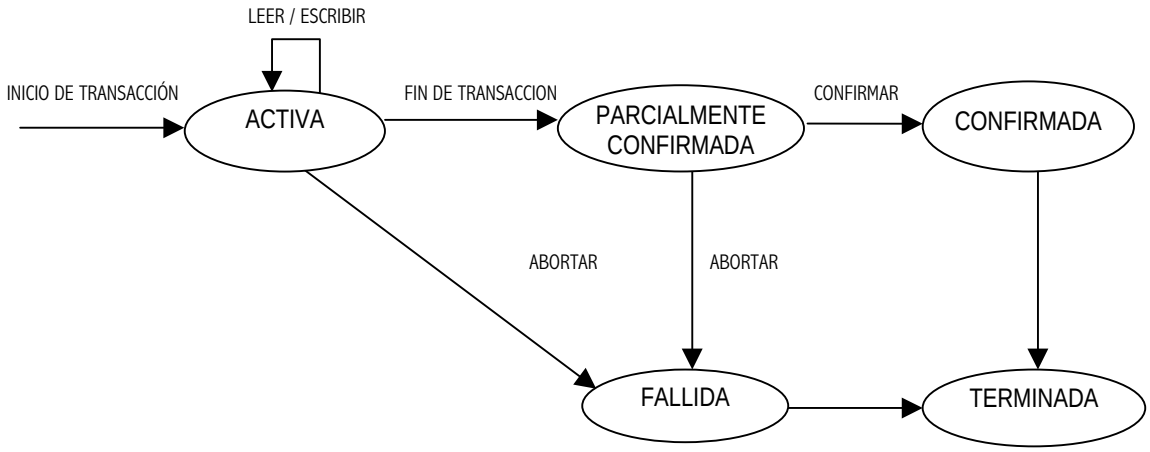
  Similar a ABORTAR, pero se aplica a una sola operación y no a una transacción completa.

- REHACER

  Especifica que algunas de las operaciones realizadas por una transacción deben repetirse, para asegurar que todas las operaciones realizadas por una transacción que ha sido CONFIRMADA se hayan aplicado con éxito a la BD (es decir, los cambios hayan quedado grabados físicamente en disco).

ESTADOS DE UNA TRANSACCIÓN

El siguiente es el diagrama de transición de estados para la ejecución de transacciones:



Una transacción entra en el estado ACTIVA justo después de iniciar su ejecución y, en este estado, puede realizar operaciones LEER y ESCRIBIR.

Cuando la transacción finaliza, pasa al estado PARCIALMENTE CONFIRMADA. En este punto, el Subsistema de Control de Concurrencia puede efectuar verificaciones para asegurar que la transacción no interfiera con otras transacciones en ejecución. Además, el Subsistema de Recuperación puede anotar qué operaciones (qué cambios) ha realizado que la transacción en un fichero del sistema (*bitácora*[3]), con el objetivo de garantizar que los cambios realizados por la transacción terminada queden permanentes, a pesar de fallos del sistema.

Una vez realizadas con éxito ambas verificaciones, la transacción ha llegado a su punto de confirmación[4] y pasa al estado CONFIRMADA (ha concluido su ejecución con éxito).

Si una de las verificaciones falla o la transacción se aborta mientras está en estado ACTIVA, pasa al estado FALLIDA. En este caso, es posible que la transacción deba ser cancelada (anulada, revertida, abortada) para anular los efectos de sus operaciones ESCRIBIR sobre la BD.

El estado TERMINADA indica que la transacción ha abandonado el sistema.

Las transacciones fallidas (abortadas) pueden ser reiniciadas posteriormente, ya sea de forma automática por parte del sistema, o bien sea el usuario el que las reintroduzca como si fueran nuevas transacciones.

[3] Estudiaremos la *bitácora del sistema* en el tema de Recuperación de Caídas en Sistemas de Base de Datos.
[4] Los *puntos de confirmación* también se verán posteriormente con más detalle.

Apéndice

Oracle7: Comandos de Control de Transacciones

Los comandos de control de transacciones manejan de forma explícita los cambios realizados por comandos DML (*Data Manipulation Language*) son los siguientes:

Comando	Propósito
COMMIT	Hacer permanentes los cambios realizados por las sentencias ejecutadas en la transacción actual, y establecer el comienzo de una nueva transacción.
ROLLBACK	Deshacer todos los cambios realizados desde el comienzo de la transacción actual o desde un *savepoint*.
SAVEPOINT	Establecer un punto hasta el cual se podrá deshacer cambios.
SET TRANSACTION	Establecer propiedades para la transacción actual.

Todos ellos, excepto ciertas formas de los comandos COMMIT y ROLLBACK, son soportados por PL/SQL. Nos centraremos en la sentencia SAVEPOINT.

SAVEPOINT

Propósito: Identificar un punto en una transacción hasta el cual se podrá hacer *rollback* posteriormente.

Sintaxis: SAVEPOINT savepoint

Donde savepoint es el nombre del *savepoint* que se está creando.

Notas de Uso:

Los *savepoints* se utilizan con el comando ROLLBACK para deshacer **porciones** de la transacción actual.

Los *savepoints* son útiles en programas interactivos, porque se puede crear y nombrar pasos intermedios de un programa. Esto permite mayor control sobre programas largos y complejos. Por ejemplo, es posible usar *savepoints* a lo largo de una serie de actualizaciones (*updates*) largas y complejas, para que si se comete un error, no se tenga que rehacer cada una de las sentencias.

Los *savepoints* son útiles en programas de aplicación de manera similar. Si un programa contiene varios subprogramas, se puede crear un *savepoint* antes del comienzo de cada subprograma. Si falla algún subprograma, es fácil volver al estado anterior de los datos antes de que empezara el subprograma (haciendo *rollback* hasta el *savepoint* adecuado), y entonces re-ejecutar el mismo una vez revisados los parámetros o realizada alguna acción de recuperación.

El nombre de un *savepoint* debe ser único dentro de una transacción dada. Si se crea un segundo *savepoint* con el mismo identificador que un *savepoint* anterior, éste último es eliminado. Después de crear un *savepoint*, se puede continuar el procesamiento, confirmar (*commit*) el trabajo, anular (*rollback*) la transacción completa, o bien hacer *rollback* hasta el *savepoint*.

Ejemplo

Tras actualizar el salario de los empleados BLAKE y CLARK, comprobamos que el salario total de la compañía excede de 20000$, lo cual no es posible, así que reintroducimos el salario de CLARK:

```
UPDATE emp SET sal = 2000 WHERE ename = 'BLAKE' ;
SAVEPOINT blake_sal ;
UPDATE emp SET sal = 1500 WHERE ename = 'CLARK' ;
SAVEPOINT clark_sal ;
SELECT SUM(sal) FROM emp ;
ROLLBACK TO SAVEPOINT blake_sal ;
UPDATE emp SET sal = 1300 WHERE ename = 'CLARK' ;
COMMIT ;
```

Oracle7: Notas acerca del Control de Transacciones Implícito

- Toda sentencia DML realiza un COMMIT antes de ejecutarse y otro después de ejecutarse (si termina con éxito).
  Estas operaciones son:
  - CREATE, ALTER y DROP (creación, alteración y eliminación de usuarios, tablas, vistas, roles, etc.)
  - RENAME (cambio de nombres de elementos del esquema)
  - GRANT y REVOKE (concesión y revocación de privilegios y roles)
  - COMMENT (añadir comentarios al diccionario de datos)
  - ANALYZE (recopilar estadísticas sobre elementos del esquema, validar estructuras de elementos, listar filas encadenadas dentro de los elementos)
  - AUDIT, NO AUDIT (activar y desactivar opciones de auditoría)
  - TRUNCATE (borrado del contenido de elementos del esquema sin eliminar la estructura de los mismos)

- Si no existe ninguna transacción en progreso (se acaba de comenzar una nueva sesión con el *SQL*Plus*, o se acaba de realizar un COMMIT o ROLLBACK explícito, o se acaba de ejecutar una sentencia DDL), al ejecutar una sentencia (digamos X) DML (INSERT, UPDATE, DELETE, SELECT) se inicia una transacción.
  Si se realizan más sentencias DML, éstas se ejecutan dentro de esa misma transacción.
  Si después se realiza un COMMIT, todas esas sentencias DML (desde la X en adelante hasta este momento) se confirmarán en disco (esto también ocurre si se sale del sistema (del *SQL*Plus*) con un EXIT o QUIT).
  Si se realiza un ROLLBACK, todas las sentencias DML serán anuladas, es decir, la base de datos queda tal y como estaba antes de realizar esa primera sentencia DML a la que llamamos X.

- Si para salir de *SQL*Plus* se cierra la ventana directamente, sin realizar EXIT o QUIT, el SGBD Oracle entiende "*terminación anormal de transacción*" y realiza un ROLLBACK de todas las sentencias DML realizadas desde el último COMMIT.