

BASES DE Datos

• Catherine M. Ricardo

**Mc
Graw
Hill**

10.1 Propiedades de las transacciones

No importa el cuidado con que se diseñe y cree una base de datos, es fácil que se dañe o destruya, a menos que haya controles apropiados de la concurrencia y técnicas de recuperación. **Recuperación** de la base de datos es el proceso de restaurarla a su estado correcto en caso de falla. El **control de la concurrencia** es la capacidad de administrar procesos simultáneos que acceden a la base de datos sin que interfieran entre sí. Ambas medidas son necesarias para impedir que los datos sean inconsistentes o se pierdan.

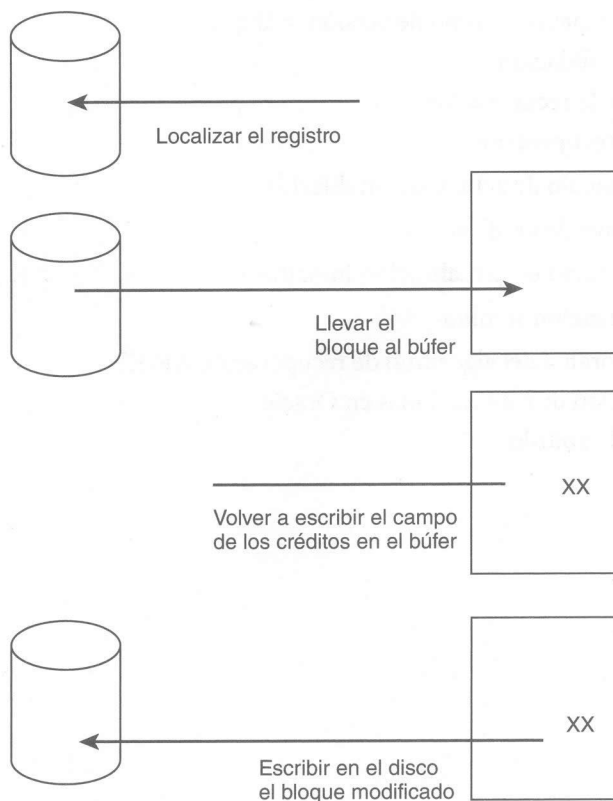
El concepto de **transacción** es fundamental para entender tanto la recuperación como el control de la concurrencia. Una transacción se puede considerar una unidad lógica de trabajo en la base de datos. Puede ser un programa completo, fragmento de programa o comando único. Involucra cualquier número de operaciones en la base de datos. Para la base de datos University se usará el esquema relacional siguiente:

```
Student (stuId, lastName, firstName, major, credits)
Faculty (facId, facname, department, rank)
Class (classNumber, facId, schedule, room)
Enroll (classNumber, stuId, grade)
```

Una transacción sencilla ejecutada en esta base de datos podría actualizar el número de créditos en un registro *Student*, dado el *stuId*. Esta tarea implica localizar en el disco el bloque que contiene el registro *Student* deseado, llevar el bloque al búfer, volver a escribir el valor del campo *credits* en el registro del búfer y, finalmente, escribir el bloque actualizado en el disco. La figura 10.1 resume los pasos de esta transacción. Una transacción más complicada sería cambiar el *stuId* asignado a un estudiante en particular. Es obvio que se necesita localizar el registro *Student* apropiado, para llevar su bloque al búfer para actua-

FIGURA 10.1

Etapas en una transacción simple



lizar el `stuId` en el búfer y escribir la actualización en el disco, igual que antes, pero también es necesario encontrar todos los registros `Enroll` que tengan el antiguo `stuId` para actualizarlos. Si estas actualizaciones no se hicieran se tendría un estado inconsistente de la base de datos, estado en que los datos son contradictorios. Una transacción siempre debe llevar a la base de datos de un estado consistente a otro consistente. Mientras la transacción esté en progreso, es permisible tener un estado temporal inconsistente. Por ejemplo, durante la actualización de `stuId` habrá un momento en que una ocurrencia de `stuId` contenga el valor nuevo y otra el viejo. Sin embargo, al final de la transacción, todas las ocurrencias coincidirán. Una transacción es el conjunto de operaciones necesarias para llevar a cabo una unidad lógica de trabajo, a fin de llevar a la base de datos a un nuevo estado consistente. La transacción es un proceso atómico, una sola unidad del "todo o nada". No se permite ejecutar sólo una parte de una transacción, se ejecutan todas las operaciones de la transacción o ninguna, ya que una transacción parcial dejaría la base de datos en un estado inconsistente.

Hay dos formas de finalizar o terminar una transacción. Si se ejecuta hasta el final con éxito, se dice que la transacción fue **comprometida (commit)**, y la base de datos es llevada a un nuevo estado consistente. La otra posibilidad es que la transacción no se ejecute con éxito. En este caso, la transacción es **abortada**. Si se aborta una transacción, es esencial que la base de datos se restaure al estado consistente en que estaba antes de comenzar la transacción. Es decir, dicha transacción fue **deshecha**, esto quiere decir que se retrocede y al mismo tiempo se deshace (**rollback**) cada una de las operaciones realizadas por la transacción hasta su inicio. Una transacción comprometida no se puede abortar. Si se decide que la transacción comprometida fue un error, se debe ejecutar otra **transacción compensadora** para revertir sus efectos. Sin embargo, una transacción abortada que haya sido deshecha puede ser reiniciada en algún momento futuro y, en función de cuál haya sido la causa de la falla, podría ejecutarse exitosamente y ser comprometida. La figura 10.2 ilustra los estados posibles de una transacción.

Por lo general, es responsabilidad del programador identificar el comienzo y final de cada transacción, toda vez que el sistema de administración de la base de datos no es capaz de determinar cuáles actualizaciones constituyen una transacción única. En ciertos DML existen las palabras `BEGIN TRANSACTION`, `END TRANSACTION`, `COMMIT`, `ABORT` y `ROLLBACK`, para delimitar transacciones. Si no se utilizan estos delimitadores, por lo general se considera a todo el programa como una transacción única, y el sistema ejecuta de manera automática la instrucción `COMMIT` cuando el programa finaliza correctamente, y `ROLLBACK` si no lo hace. El estado activo de la transacción comienza con el enunciado `BEGIN TRANSACTION` y continúa hasta que el programa de aplicación aborta o concluye con éxito, y en este último caso se llega a `END TRANSACTION`, y el DBMS (sistema de gestión de base de datos) se prepara para comprobar que la transacción fue comprometida,

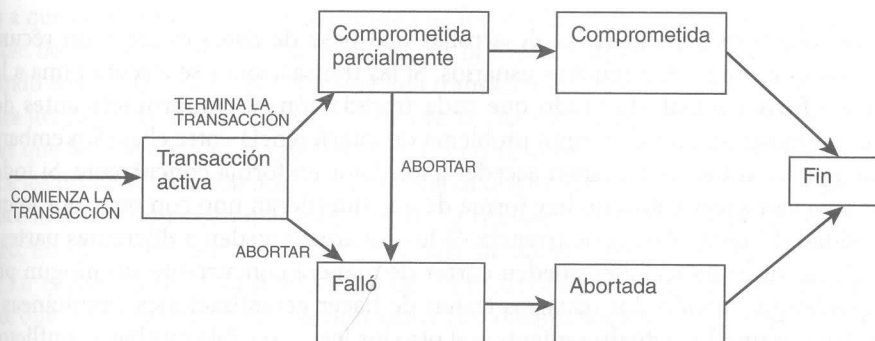


FIGURA 10.2

Diagrama del estado de la transacción

en cuyo caso se ejecuta COMMIT. Durante esta etapa de comprometimiento parcial, el DBMS verifica que la transacción no viole los protocolos de control de concurrencia que se estudian en la sección 10.4, o alguna restricción, y que el sistema sea capaz de hacer los cambios requeridos en la base de datos. Si no surgen problemas, la transacción se compromete. Por otro lado, si ocurriera un error fatal mientras la transacción está activa, se etiqueta como fracaso y se aborta. Si se hubieran hecho cualesquiera actualizaciones a la base de datos, se deshacen, es decir se realiza un ROLLBACK a la transacción. Aun después de que la transacción haya terminado con éxito, si hubiera algún problema con el control de concurrencia, integridad o falla del sistema mientras se está en el estado de comprometimiento parcial, es posible que la transacción se etiquete como fracaso y se aborte.

Para garantizar que la base de datos mantiene un estado correcto a pesar de una falla de concurrencia o del sistema, todas las transacciones deben mostrar cuatro propiedades importantes, por lo general llamadas **ACID**, que es un acrónimo en inglés de las propiedades siguientes:

- **Atomicidad.** La transacción es una sola unidad de “todo o nada”. Se ejecuta todo el conjunto de acciones o ninguna. Para garantizar esta propiedad, el DBMS debe ser capaz de cancelar transacciones que no terminen con éxito, y anular sus efectos sobre la base de datos. El subsistema de recuperación del DBMS mantiene un **registro**, llamado **bitácora**, de todas las transacciones escritas en la base de datos, el cual se utiliza al deshacer la transacción.
- **Consistencia.** El usuario es responsable de asegurar que su transacción, si se ejecutara por sí sola, deje a la base de datos en un estado consistente. Es trabajo del subsistema de control de concurrencia del DBMS garantizar la consistencia cuando se ejecutan al mismo tiempo transacciones múltiples.
- **Aislamiento.** Es posible que varias transacciones se ejecuten al mismo tiempo con sus operaciones intercaladas. La propiedad de aislamiento requiere que el efecto final sea como si las transacciones se hubieran ejecutado una después de otra en vez de ejecutarse en forma concurrente. Como cada transacción de manera aislada deja a la base de datos en un estado consistente, el resultado en conjunto de todas las transacciones también será un estado consistente. El sistema de control de concurrencia tiene que garantizar el aislamiento.
- **Durabilidad.** Si una transacción ha sido comprometida, el DBMS debe asegurar que sus efectos se registren de manera permanente en la base de datos, aun si el sistema fallara antes de que se hubieran escrito en la base de datos los registros involucrados por la transacción. El subsistema de recuperación es responsable de garantizar la persistencia de los datos, para lo que utiliza la bitácora de las transacciones.

10.2 Necesidad del control de la concurrencia

Uno de los objetivos principales al desarrollar una base de datos es crear un recurso de información que compartan muchos usuarios. Si las transacciones se ejecutan una a la vez, es decir, **en forma serial**, de modo que cada transacción se comprometa antes de que comience la siguiente, no hay ningún problema de interferencia entre ellas. Sin embargo, es frecuente que los usuarios necesiten acceder a los datos en forma concurrente. Si todos los usuarios sólo van a leer datos, no hay forma de que interfieran uno con otro, por lo que no hay necesidad del control de concurrencia. Si los usuarios acceden a diferentes partes de la base de datos, sus transacciones pueden correr de manera concurrente sin ningún problema. Sin embargo, cuando dos usuarios tratan de hacer actualizaciones simultáneas a los mismos datos, o uno los actualiza mientras el otro los lee, es posible que haya conflictos.