- **Active**, the initial state; the transaction stays in this state while it is executing.
- **Partially committed**, after the final statement has been executed.
- **Failed**, after the discovery that normal execution can no longer proceed.
- **Aborted**, after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- **Committed**, after successful completion.

The state diagram corresponding to a transaction appears in Figure 17.1. We say that a transaction has committed only if it has entered the committed state. Similarly, we say that a transaction has aborted only if it has entered the aborted state. A transaction is said to have **terminated** if it has either committed or aborted.

A transaction starts in the active state. When it finishes its final statement, it enters the partially committed state. At this point, the transaction has completed its execution, but it is still possible that it may have to be aborted, since the actual output may still be temporarily residing in main memory, and thus a hardware failure may preclude its successful completion.

The database system then writes out enough information to disk that, even in the event of a failure, the updates performed by the transaction can be re-created when the system restarts after the failure. When the last of this information is written out, the transaction enters the committed state.

As mentioned earlier, we assume for now that failures do not result in loss of data on disk. Chapter 19 discusses techniques to deal with loss of data on disk.

A transaction enters the failed state after the system determines that the transaction can no longer proceed with its normal execution (e.g., because of hardware or logical errors). Such a transaction must be rolled back. Then, it enters the aborted state. At this point, the system has two options:

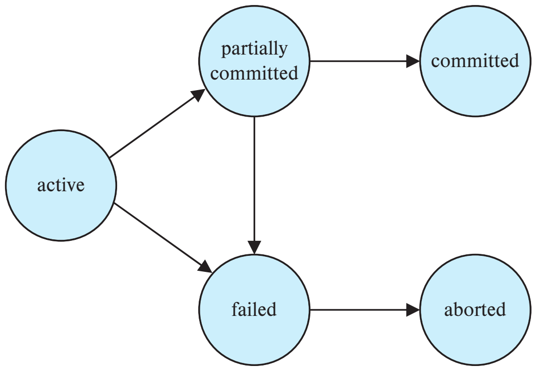

Figure 17.1 State diagram of a transaction.