A number of extensions were suggested to the transactions as suggested by Gray. [Moss, 1985] suggested the concept of nested transactions as we have discussed them in this section. Mechanisms for the implementation of nested transactions are still not widely available in either databases or object-oriented middleware, which is why we have refrained from discussing them further.

The transactions that we have discussed here are very suitable when the overall amount of processing can be completed in a short period of time. Transactions have also been suggested for long-lived tasks, such as designing a circuit or a software architecture. These long transactions give up one ACID property or another in order to be suitable for collaborative and interactive long-lived activities. A good overview of these non-standard transactions and their application to software engineering is provided by [Barghouti and Kaiser, 1991]. The survey includes CAD transactions [Bancilhon et al., 1985], design transactions [Katz, 1984] and split–join transactions [Kaiser, 1990].

[Bernstein et al., 1987] wrote the definitive textbook on database concurrency control. The book includes a very thorough discussion of the different locking techniques that we could only briefly discuss in this chapter. The Concurrency Control service of CORBA is discussed in Chapter 7 of [Object Management Group, 1996]. Java provides built-in primitives for concurrency control. Java operations can be declared as synchronized, which means that only one of these synchronized operations can be executed at any point in time. The locking mechanism that is used to implement synchronization, however, does not distinguish different locking modes and thus may be overly restrictive. A good discussion of the concurrency primitives of Java is provided by [Magee and Kramer, 1999] and [Lea, 1997].

The discussion of the services that are available to implement distributed object transaction had to be rather brief in this chapter, as we felt it more important to focus on the underlying principles. The CORBA Object Transaction service is defined in detail in Chapter 10 of [Object Management Group, 1996]. A professional text book that covers Microsoft's Transaction Service rather well is [Hillier, 1998]. The Java architecture for transaction management is defined in the Java Transaction API [Cheung and Matena, 1999]. It relies on the Java Transaction Service, a Java binding to the CORBA Transaction Service. The Java Transaction Service is defined in [Cheung, 1999]. [Sun, 1999] specifies transactions for Jini, the Java-based middleware for physically mobile devices. Again, those transactions are implemented using a two-phase commit protocol.

Distributed transactions do not necessarily have to be implemented in an object-oriented way. *Transaction monitors* support the implementation of distributed transactions between non-object-oriented programs. Examples of these transaction monitors include CICS from IBM and Tuxedo from BEA. CICS is extensively described by [Hudders, 1994]. A comprehensive introduction to Tuxedo is given by [Andrade, 1996]. These transaction monitors are also often used to implement distributed object transaction services. The integration of both transaction monitors and distributed object transaction services with database systems relies on the Open Distributed Transaction Processing Standard of the Open Group. The XA Protocol for database transaction control is defined in [X/Open Group, 1994].