

Chapter 7 (Cont.)

Transaction Management and Concurrency Control

In this chapter, you will learn:

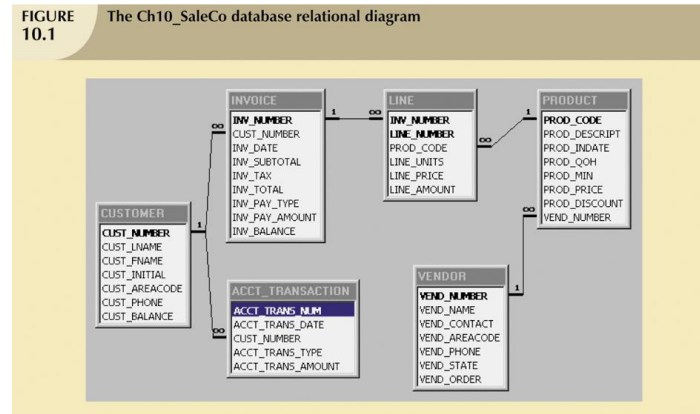
- What a database transaction is and what its properties are
- What concurrency control is and what role it plays in maintaining the database's integrity
- What locking methods are and how they work
- How stamping methods are used for concurrency control

In this chapter, you will learn (continued):

- How optimistic methods are used for concurrency control
- How database recovery management is used to maintain database integrity

What is a Transaction?

FIGURE 10.1 The Ch10_SaleCo database relational diagram



What is a Transaction? (continued)

- Any action that reads from and/or writes to a database may consist of:
 - Simple SELECT statement to generate list of table contents
 - Series of related UPDATE statements to change values of attributes in various tables
 - Series of INSERT statements to add rows to one or more tables
 - Combination of SELECT, UPDATE, and INSERT statements

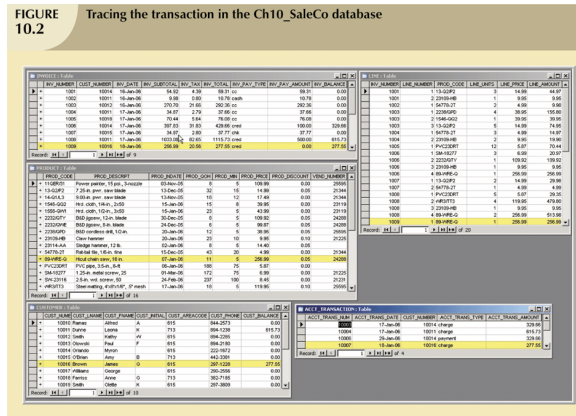
What is a Transaction? (continued)

- Transaction is logical unit of work that must be either entirely completed or aborted
- Successful transaction changes database from one consistent state to another
 - One in which all data integrity constraints are satisfied
- Most real-world database transactions are formed by two or more database requests
 - Equivalent of a single SQL statement in an application program or transaction

Evaluating Transaction Results

- Not all transactions update database
- SQL code represents a transaction because database was accessed
- Improper or incomplete transactions can have devastating effect on database integrity
 - Some DBMSs provide means by which user can define enforceable constraints
 - Other integrity rules are enforced automatically by the DBMS

Evaluating Transaction Results (continued)



Transaction Properties

- Atomicity
 - Requires that all operations (SQL requests) of a transaction be completed
- Consistency
 - Indicates the permanence of database's consistent state

Transaction Properties (continued)

- Isolation
 - Data used during execution of a transaction cannot be used by second transaction until first one is completed
- Durability
 - Indicates permanence of database's consistent state Isolation

Transaction Properties (continued)

- Serializability
 - Ensures that concurrent execution of several transactions yields consistent results

Transaction Management with SQL

- ANSI has defined standards that govern SQL database transactions
- Transaction support is provided by two SQL statements: COMMIT and ROLLBACK

Transaction Management with SQL (continued)

- ANSI standards require that, when a transaction sequence is initiated by a user or an application program, it must continue through all succeeding SQL statements until one of four events occurs
 - COMMIT statement is reached
 - ROLLBACK statement is reached
 - End of program is reached
 - Program is abnormally terminated

The Transaction Log

- Transaction log stores:
 - A record for the beginning of transaction
 - For each transaction component (SQL statement):
 - Type of operation being performed (update, delete, insert)
 - Names of objects affected by transaction
 - “Before” and “after” values for updated fields
 - Pointers to previous and next transaction log entries for the same transaction
 - Ending (COMMIT) of the transaction

The Transaction Log (continued)

TABLE 10.1 A Transaction Log

TRL_ID	TRX_NUM	PREV_PTR	NEXT_PTR	OPERATION	TABLE	ROW_ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	****Start Transaction				
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	525.75	615.73
365	101	363	Null	COMMIT	**** End of Transaction				



TRL_ID = Transaction log record ID **PTR** = Pointer to a transaction log record ID
TRX_NUM = Transaction number
 (Note: The transaction number is automatically assigned by the DBMS.)

Concurrency Control

- Coordination of simultaneous transaction execution in a multiprocessing database system
- Objective is to ensure serializability of transactions in a multiuser database environment

Concurrency Control (continued)

- Simultaneous execution of transactions over a shared database can create several data integrity and consistency problems
 - Lost updates
 - Uncommitted data
 - Inconsistent retrievals

Lost Updates

TABLE 10.2 Normal Execution of Two Transactions

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	$\text{PROD_QOH} = 35 + 100$	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH	135
5	T2	$\text{PROD_QOH} = 135 - 30$	
6	T2	Write PROD_QOH	105

Lost Updates (continued)

TABLE 10.3 Lost Updates

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T2	Read PROD_QOH	35
3	T1	$PROD_QOH = 35 + 100$	
4	T2	$PROD_QOH = 35 - 30$	
5	T1	Write PROD_QOH (Lost update)	135
6	T2	Write PROD_QOH	5

Uncommitted Data

TABLE 10.4 Correct Execution of Two Transactions

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	$\text{PROD_QOH} = 35 + 100$	
3	T1	Write PROD_QOH	135
4	T1	**** ROLLBACK ****	35
5	T2	Read PROD_QOH	35
6	T2	$\text{PROD_QOH} = 35 - 30$	
7	T2	Write PROD_QOH	5

Uncommitted Data (continued)

TABLE
10.5

An Uncommitted Data Problem

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	$PROD_QOH = 35 + 100$	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH (Read uncommitted data)	135
5	T2	$PROD_QOH = 135 - 30$	
6	T1	***** ROLLBACK *****	35
7	T2	Write PROD_QOH	105

Inconsistent Retrievals

TABLE 10.6 Retrieval During Update

TRANSACTION T1		TRANSACTION T2	
SELECT	SUM(PROD_QOH)	UPDATE	PRODUCT
FROM	PRODUCT	SET	PROD_QOH = PROD_QOH + 10
		WHERE	PROD_CODE = '1546-QQ2'
		UPDATE	PRODUCT
		SET	PROD_QOH = PROD_QOH - 10
		WHERE	PROD_CODE = '1558-QW1'
		COMMIT;	

Inconsistent Retrievals (continued)

TABLE 10.7 Transaction Results: Data Entry Correction

	BEFORE	AFTER
PROD_CODE	PROD_QOH	PROD_QOH
11QER/31	8	8
13-Q2/P2	32	32
1546-QQ2	15	(15 + 10) → 25
1558-QW1	23	(23 - 10) → 13
2232-QTY	8	8
2232-QWE	6	6
Total	92	92

Inconsistent Retrievals (continued)

TABLE 10.8 Inconsistent Retrievals

TIME	TRANSACTION	ACTION	VALUE	TOTAL
1	T1	Read PROD_QOH for PROD_CODE = '11QER/31'	8	8
2	T1	Read PROD_QOH for PROD_CODE = '13-Q2/P2'	32	40
3	T2	Read PROD_QOH for PROD_CODE = '1546-QQ2'	15	
4	T2	PROD_QOH = 15 + 10		
5	T2	Write PROD_QOH for PROD_CODE = '1546-QQ2'	25	
6	T1	Read PROD_QOH for PROD_CODE = '1546-QQ2'	25	(After) 65
7	T1	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	(Before) 88
8	T2	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	
9	T2	PROD_QOH = 23 - 10		
10	T2	Write PROD_QOH for PROD_CODE = '1558-QW1'	13	
11	T2	***** COMMIT *****		
12	T1	Read PROD_QOH for PROD_CODE = '2232-QTY'	8	96
13	T1	Read PROD_QOH for PROD_CODE = '2232-QWE'	6	102

The Scheduler

- Special DBMS program
 - Purpose is to establish order of operations within which concurrent transactions are executed
- Interleaves execution of database operations to ensure serializability and isolation of transactions

The Scheduler (continued)

- Bases its actions on concurrency control algorithms
- Ensures computer's central processing unit (CPU) is used efficiently
- Facilitates data isolation to ensure that two transactions do not update same data element at same time

The Scheduler (continued)

TABLE
10.9

Read/Write Conflict Scenarios: Conflicting Database Operations Matrix

	TRANSACTIONS		RESULT
	T1	T2	
Operations	Read	Read	No conflict
	Read	Write	Conflict
	Write	Read	Conflict
	Write	Write	Conflict

Concurrency Control with Locking Methods

- Lock
 - Guarantees exclusive use of a data item to a current transaction
 - Required to prevent another transaction from reading inconsistent data
- Lock manager
 - Responsible for assigning and policing the locks used by transactions

Lock Granularity

- Indicates level of lock use
- Locking can take place at following levels:
 - Database
 - Table
 - Page
 - Row
 - Field (attribute)

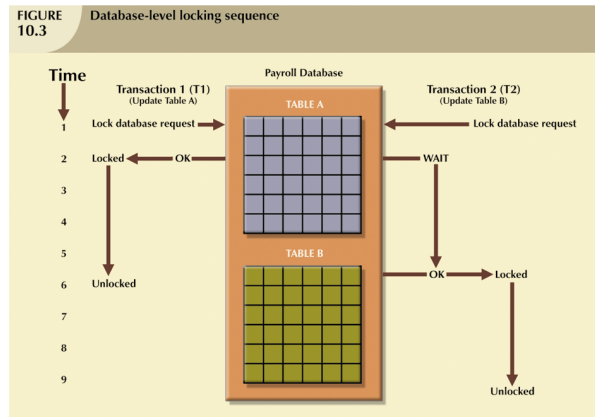
Lock Granularity (continued)

- Database-level lock
 - Entire database is locked
- Table-level lock
 - Entire table is locked
- Page-level lock
 - Entire diskpage is locked

Lock Granularity (continued)

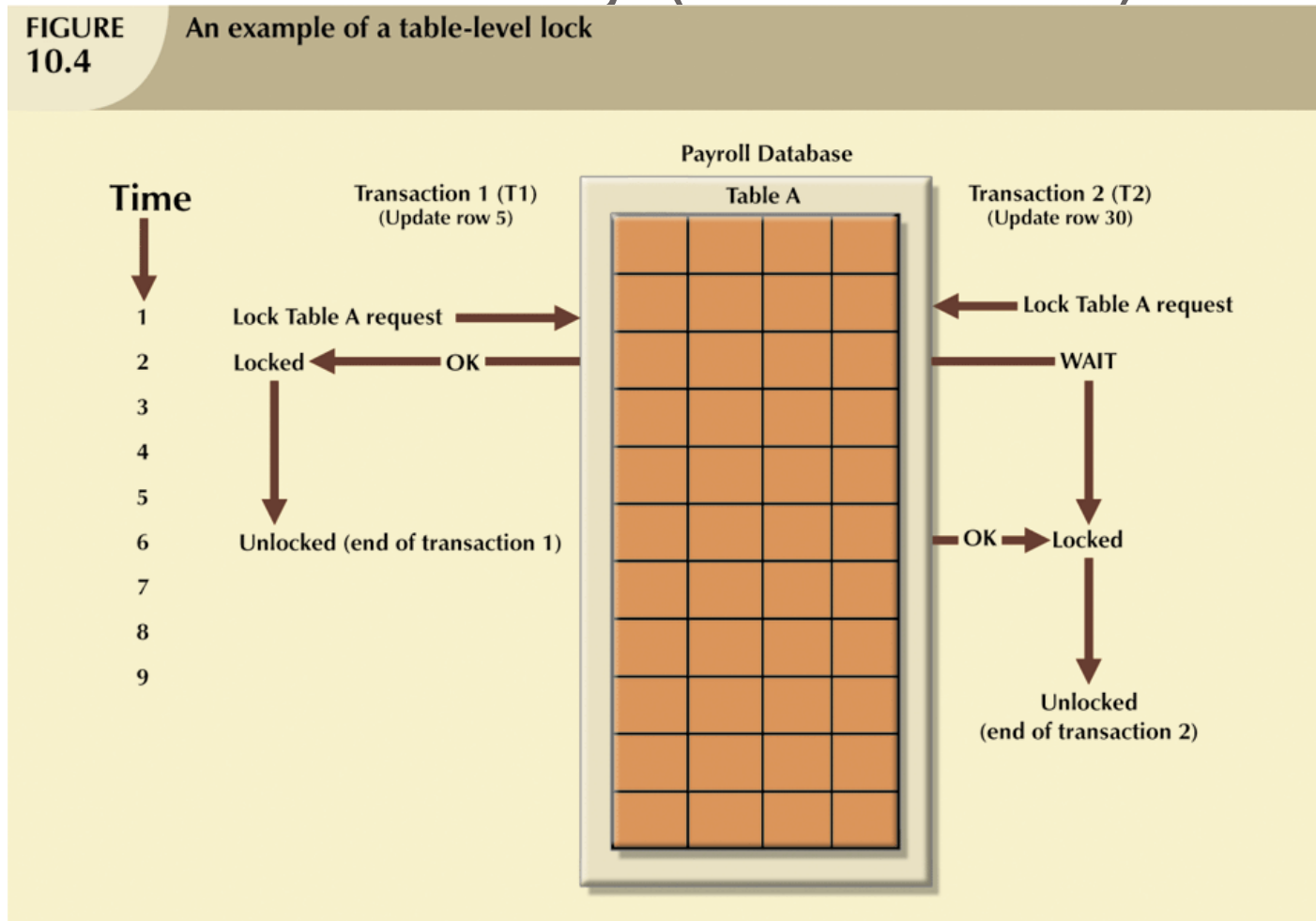
- Row-level lock
 - Allows concurrent transactions to access different rows of same table, even if rows are located on same page
- Field-level lock
 - Allows concurrent transactions to access same row, as long as they require use of different fields (attributes) within that row

Lock Granularity (continued)

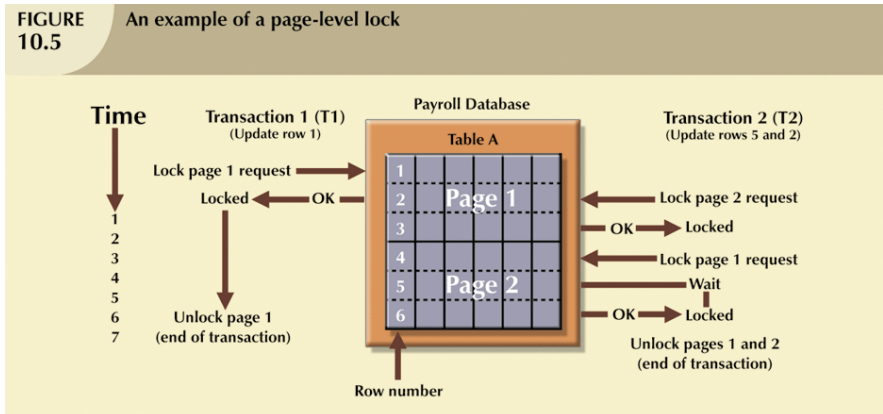


Lock Granularity (continued)

FIGURE 10.4 An example of a table-level lock

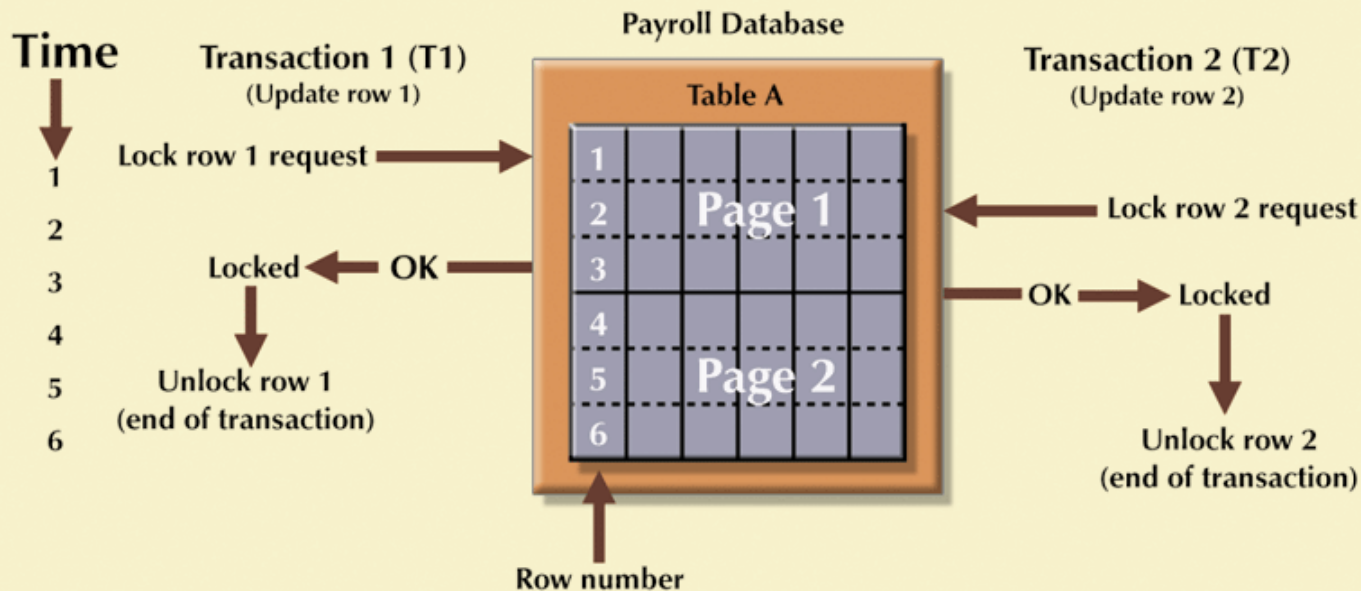


Lock Granularity (continued)



Lock Granularity (continued)

FIGURE 10.6 An example of a row-level lock



Lock Types

- Binary lock
 - Has only two states: locked (1) or unlocked (0)
- Exclusive lock
 - Access is specifically reserved for transaction that locked object
 - Must be used when potential for conflict exists
- Shared lock
 - Concurrent transactions are granted Read access on basis of a common lock

Lock Types (continued)

TABLE 10.10 An Example of a Binary Lock

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Lock PRODUCT	
2	T1	Read PROD_QOH	15
3	T1	$PROD_QOH = 15 + 10$	
4	T1	Write PROD_QOH	25
5	T1	Unlock PRODUCT	
6	T2	Lock PRODUCT	
7	T2	Read PROD_QOH	23
8	T2	$PROD_QOH = 23 - 10$	
9	T2	Write PROD_QOH	13
10	T2	Unlock PRODUCT	

Two-Phase Locking to Ensure Serializability

- Defines how transactions acquire and relinquish locks
- Guarantees serializability, but it does not prevent deadlocks
 - Growing phase - Transaction acquires all required locks without unlocking any data
 - Shrinking phase - Transaction releases all locks and cannot obtain any new lock

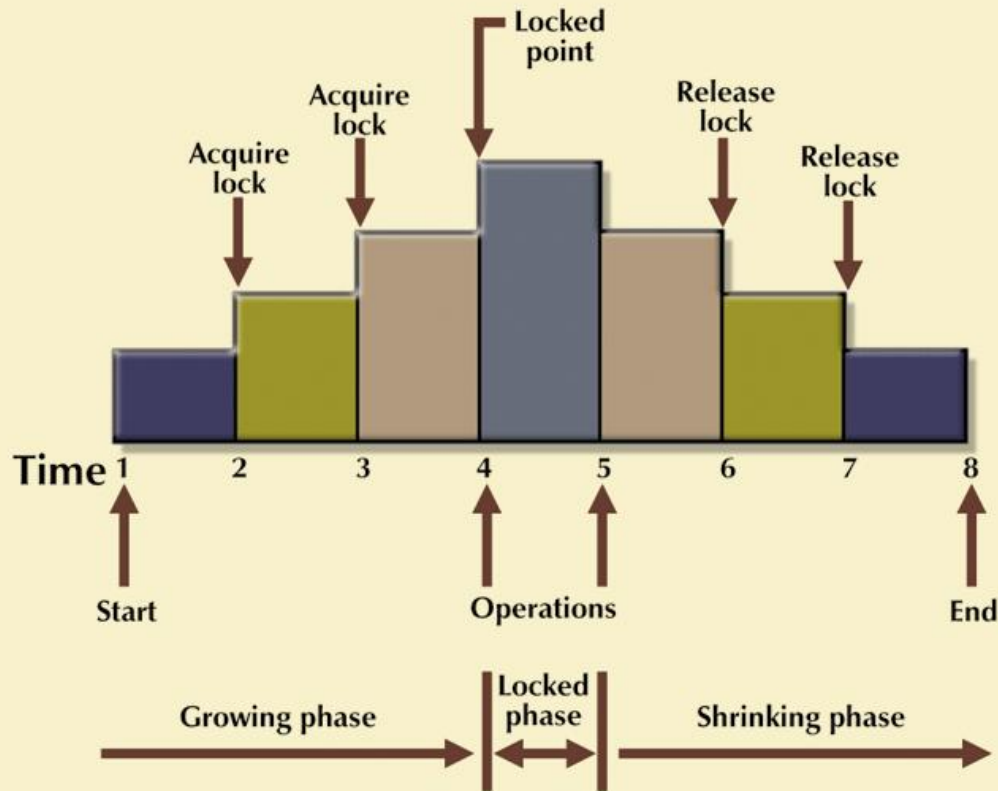
Two-Phase Locking to Ensure Serializability (continued)

- Governed by the following rules:
 - Two transactions cannot have conflicting locks
 - No unlock operation can precede a lock operation in the same transaction
 - No data are affected until all locks are obtained—that is, until transaction is in its locked point

Two-Phase Locking to Ensure Serializability (continued)

FIGURE
10.7

Two-phase locking protocol



Deadlocks

- Condition that occurs when two transactions wait for each other to unlock data
- Possible only if one of the transactions wants to obtain an exclusive lock on a data item
 - No deadlock condition can exist among shared locks

Deadlocks (continued)

- Control through:
 - Prevention
 - Detection
 - Avoidance

Deadlocks (continued)

TABLE 10.11 How a Deadlock Condition Is Created

TIME	TRANSACTION	REPLY	LOCK STATUS	
0			Data X	Data Y
1	T1:LOCK(X)	OK	Unlocked	Unlocked
2	T2: LOCK(Y)	OK	Locked	Unlocked
3	T1:LOCK(Y)	WAIT	Locked	Locked
4	T2:LOCK(X)	WAIT	Locked	Locked
5	T1:LOCK(Y)	WAIT	Locked	Locked
6	T2:LOCK(X)	WAIT	Locked	Locked
7	T1:LOCK(Y)	WAIT	Locked	Locked
8	T2:LOCK(X)	WAIT	Locked	Locked
9	T1:LOCK(Y)	WAIT	Locked	Locked
...
...
...
...



Concurrency Control with Time Stamping Methods

- Assigns global unique time stamp to each transaction
- Produces explicit order in which transactions are submitted to DBMS
- Uniqueness
 - Ensures that no equal time stamp values can exist
- Monotonicity
 - Ensures that time stamp values always increase

Wait/Die and Wound/Wait Schemes

- Wait/die
 - Older transaction waits and younger is rolled back and rescheduled
- Wound/wait
 - Older transaction rolls back younger transaction and reschedules it

Wait/Die and Wound/Wait Schemes (continued)

TABLE 10.12 Wait/Die and Wound/Wait Concurrency Control Schemes

TRANSACTION REQUESTING LOCK	TRANSACTION OWNING LOCK	WAIT/DIE SCHEME	WOUND/WAIT SCHEME
T1 (11548789)	T2 (19562545)	<ul style="list-style-type: none">• T1 waits until T2 is completed and T2 releases its locks.	<ul style="list-style-type: none">• T1 preempts (rolls back) T2.• T2 is rescheduled using the same time stamp.
T2 (19562545)	T1 (11548789)	<ul style="list-style-type: none">• T2 dies (rolls back).• T2 is rescheduled using the same time stamp.	<ul style="list-style-type: none">• T2 waits until T1 is completed and T1 releases its locks.

Concurrency Control with Optimistic Methods

- Optimistic approach
 - Based on assumption that majority of database operations do not conflict
 - Does not require locking or time stamping techniques
 - Transaction is executed without restrictions until it is committed
 - Phases are read, validation, and write

Database Recovery Management

- Database recovery
 - Restores database from given state, usually inconsistent, to previously consistent state
 - Based on atomic transaction property
 - All portions of transaction must be treated as single logical unit of work, so all operations must be applied and completed to produce consistent database
 - If transaction operation cannot be completed, transaction must be aborted, and any changes to database must be rolled back (undone)

Transaction Recovery

- Makes use of deferred-write and write-through techniques
- Deferred write
 - Transaction operations do not immediately update physical database
 - Only transaction log is updated
 - Database is physically updated only after transaction reaches its commit point using transaction log information

Transaction Recovery (continued)

- Write-through
 - Database is immediately updated by transaction operations during transaction's execution, even before transaction reaches its commit point

Transaction Recovery (continued)

TABLE 10.13 A Transaction Log for Transaction Recovery Examples

TRL ID	TRX NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	**** Start Transaction				
352	101	341	363	UPDATE	PRODUCT	54778-2T	PROD_QOH	45	43
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	615.73	675.62
365	101	363	Null	COMMIT	**** End of Transaction				
397	106	Null	405	START	**** Start Transaction				
405	106	397	415	INSERT	INVOICE	1009			1009,10016, ...
415	106	405	419	INSERT	LINE	1009,1			1009,1, 89-WRE-Q,1, ...
419	106	415	427	UPDATE	PRODUCT	89-WRE-Q	PROD_QOH	12	11
423				CHECKPOINT					
427	106	419	431	UPDATE	CUSTOMER	10016	CUST_BALANCE	0.00	277.55
431	106	427	457	INSERT	ACCT_TRANSACTION	10007			1007,18-JAN-2004, ...
457	106	431	Null	COMMIT	**** End of Transaction				
521	155	Null	525	START	**** Start Transaction				
525	155	521	528	UPDATE	PRODUCT	2232/QWE	PROD_QOH	6	26
528	155	525	Null	COMMIT	**** End of Transaction				
*****C*R*A*S*H*****									

Summary

- Transaction
 - Sequence of database operations that access database
 - Represents real-world events
 - Must be logical unit of work
 - No portion of transaction can exist by itself
 - Takes database from one consistent state to another
 - One in which all data integrity constraints are satisfied

Summary (continued)

- Transactions have five main properties: atomicity, consistency, isolation, durability, and serializability
- SQL provides support for transactions through the use of two statements: COMMIT and ROLLBACK
- SQL transactions are formed by several SQL statements or database requests

Summary (continued)

- Transaction log keeps track of all transactions that modify database
- Concurrency control coordinates simultaneous execution of transactions
- Scheduler is responsible for establishing order in which concurrent transaction operations are executed

Summary (continued)

- Lock guarantees unique access to a data item by transaction
- Two types of locks can be used in database systems: binary locks and shared/exclusive locks
- Serializability of schedules is guaranteed through the use of two-phase locking

Summary (continued)

- When two or more transactions wait indefinitely for each other to release lock, they are in deadlock, or deadly embrace
- Three deadlock control techniques: prevention, detection, and avoidance

Summary (continued)

- Concurrency control with time stamping methods assigns unique time stamp to each transaction and schedules execution of conflicting transactions in time stamp order

Summary (continued)

- Concurrency control with optimistic methods assumes that the majority of database transactions do not conflict and that transactions are executed concurrently, using private copies of the data
- Database recovery restores database from given state to previous consistent state