

"Time was invented to keep everything from happening at once."



- Steven Wright

Topics

- Transaction Management
- Concurrency Control
- Group Work

Chapter 10

Transaction and Concurrency Control

Transactions

Read or Write to DB

Because of Multiple Tables
Single Transaction
Multiple I/O Requests
Considered a Single Action

Consistent Database State

Transaction

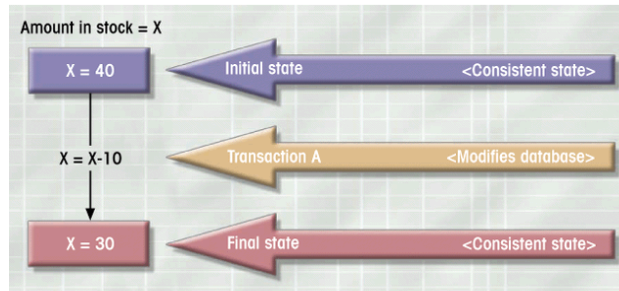


FIGURE 9.1 EXAMPLE OF A TRANSACTION

Consistent Database State

Read

DB remain in Consistent State

Write

- Sell 100 units to Customer X for 5000 dollars
- Two Updates
- Inventory, Accts Receivable
- BOTH need to be performed

Transaction Properties

Atomicity - All parts of transaction

Durability - DB in consistent state

Serializability

- Concurrent Transactions

Isolation

- Locking Records

Transaction Management

Commit

All changes written to DB

Automatic at end of program

Rollback

Changes abandoned

Database returns to consistent state

Abnormal Termination

Transaction Log

Keep track of every change

It is another database

Watch for Disk Full / Disk Crash

Redundancy / Backups

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	345TYX	PROD_QOH	243	143
363	101	352	365	UPDATE	ACCT_RECEIVABLE	60120010	ACCT_BALANCE	1200	4700
365	101	363	Null	COMMIT	**** End 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

Lost Updates

T1: Purchase 100 Units

T2: Sell 30 Units

PROD_QOH = 35

Sequence

PROD_QOH = 35

T1: Read 35

T2: Read 35

T1: Add 100, Write 135

PROD_QOH = 135

T2: Subtract 30, Write 5

PROD_QOH = 5

We've lost the 100 bought

Uncommitted Data

Rollback T1

PROD_QOH = 35

T1: Read 35

T1: Add 100, Write 135

PROD_QOH = 135

T2: Read 135, Subtract 30

T1: Rollback

T2: Write 105

PROD_QOH = 105

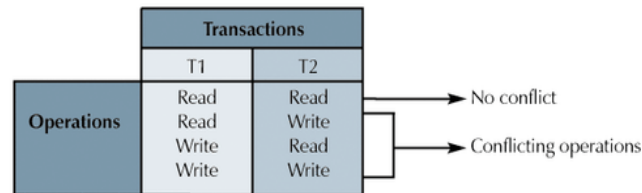
Inconsistent Results

One Transaction summarizes

While other Transactions update

The Scheduler

Ensures Serializability



Methods

Locking

Time Stamping

Optimistic

Locking

Exclusive access for current trans.

Lock

Access

Unlock

Others need to wait

Lock Manager

Granularity

Database

Table

Page (Disk Lock)

Row

Field

Example: Table Lock

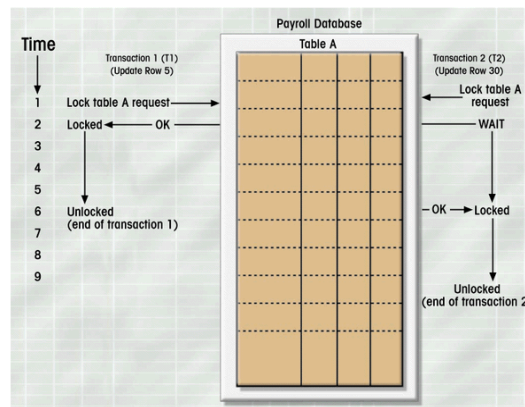


FIGURE 9.3 AN EXAMPLE OF A TABLE-LEVEL LOCK

Lock Types

Binary: Lock / Unlock

Shared - Read Only

Others may read

Exclusive - Write

No other access

Problems: Deadlocks

Dead Lock

T1 needs X & Y while T2 needs Y & X

TIME	TRANSACTION	REPLY	LOCK STATUS	
0			Data X	Data Y
			Unlocked	Unlocked
1	T1:LOCK(X)	OK	Locked	Unlocked
2	T2:LOCK(Y)	OK	Locked	Locked
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
...
...
...
...



Time Stamping

Uniqueness

No two time stamps the same

Monotonicity

No time warps

Optimistic Methods

Assumption

Most Transaction do not conflict

Few Updates

Read

Validate

Write

Database Recovery Management

Full backup

Differential Backups

Backup of Transaction Logs

Group Work

Meeting with Teacher

Let me see what you have

Design Review

End of Lesson