



C340 Concurrency: Databases Concurrency Control

Wolfgang Emmerich



Outline

- ***Motivation***
- ***Locking and Lock Compatibility***
- ***Two-Phase Locking***
- ***Hierarchical Locking***
- ***Implicit vs. Explicit Locking***
- ***CORBA Concurrency Control Service***



Motivation

- **Components of distributed systems use shared resources concurrently:**
 - *Hardware Components*
 - *Operating system resources*
 - *Databases*
 - *Objects*
- **Resources may have to be accessed in mutual exclusion.**



Motivation

- **Concurrent access and updates of resources may lead to:**
 - *lost updates*
 - *inconsistent analysis.*
- **Example for lost updates:**
 - *Cash withdrawal from ATM and concurrent*
 - *Credit of cheque.*
- **Example for inconsistent analysis:**
 - *Funds transfer between accounts of a customer*
 - *Sum of account balances (Report for Inland Revenue).*



Motivating Examples

```
class Account {
  protected:
    float balance;
  public:
    float get_balance() {return balance;};
    void debit(float amount){
      float new=balance-amount;
      balance=new;
    };
    void credit(float amount) {
      float new=balance+amount;
      balance=new;
    };
};
```

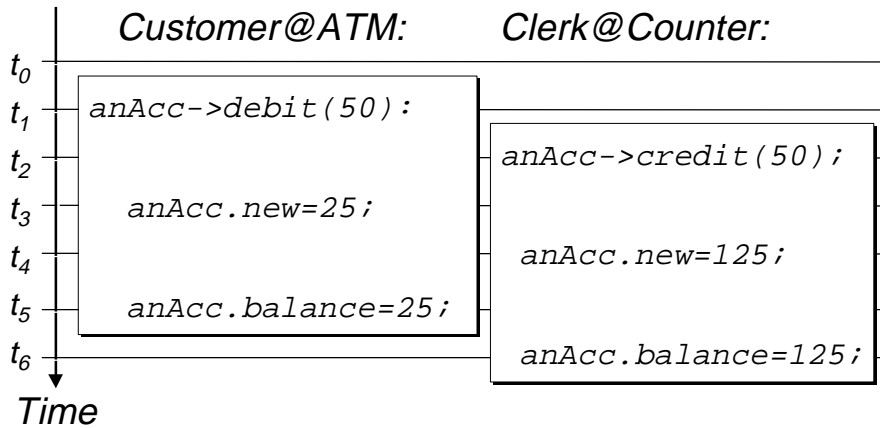
© Wolfgang Emmerich, 1998/99

5



Lost Updates

Balance of account *anAcc* at t_0 is 75



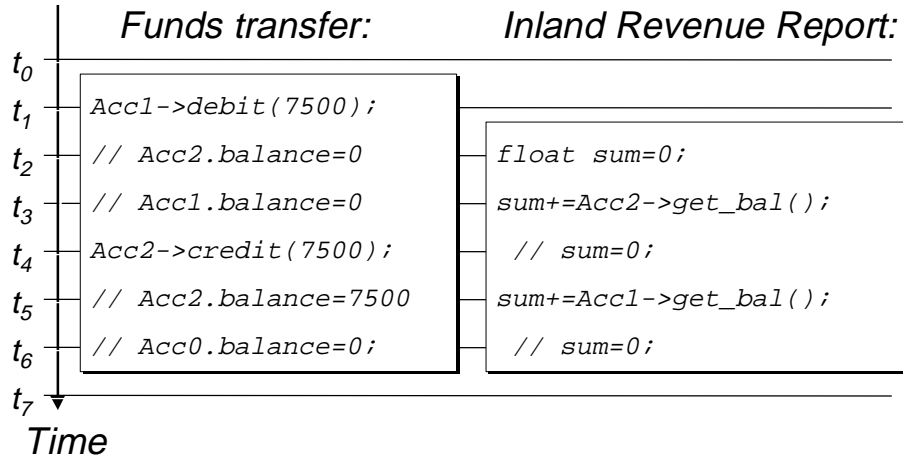
© Wolfgang Emmerich, 1998/99

6



Inconsistent Analysis

Balances at t_0 Acc1: 7500, Acc2: 0



© Wolfgang Emmerich, 1998/99

7



Two Phase Locking (2PL)

- **The most popular concurrency control technique. Used in:**
 - RDBMSs (Oracle, Ingres, Sybase, DB/2, etc.)
 - ODBMSs (O2, ObjectStore, Versant, etc.)
 - Transaction Monitors (CICS, etc)
- **Concurrent processes acquire locks on shared resources from lock manager.**
- **Lock manager grants lock if request does not conflict with already granted locks.**
- **Guarantees serialisability.**

© Wolfgang Emmerich, 1998/99

8



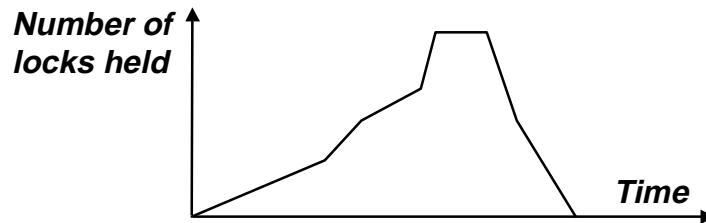
Locks

- ***A lock is a token that indicates that a process accesses a resource in a particular mode.***
- ***Minimal lock modes: read and write.***
- ***Locks are used to indicate to concurrent processes the current use of that resource.***



Locking

- ***Processes acquire locks before they access shared resources and release locks afterwards.***
- ***2PL: Processes do not acquire locks once they have released a lock.***
- ***Typical 2PL locking profile of a process:***





Lock Compatibility

- *Lock manager grants locks.*
- *Grant depends on compatibility of acquisition request with modes of already granted locks.*
- *Compatibility defined in lock compatibility matrix.*
- *Minimal lock compatibility matrix:*

	Read	Write
Read	+	-
Write	-	-



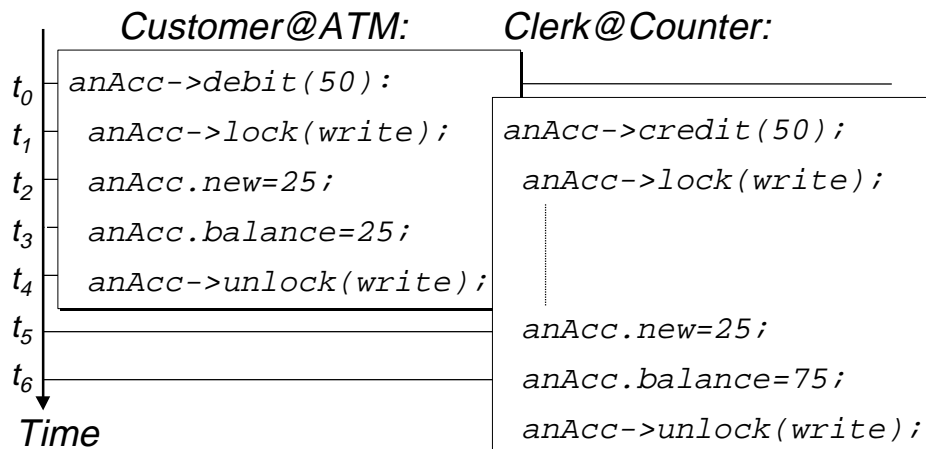
Locking Conflicts

- *Lock requests cannot be granted if incompatible locks are held by concurrent processes.*
- *This is referred to as a locking conflict.*
- *Approaches to handle conflicts:*
 - *Force requesting process to wait until conflicting locks are released.*
 - *Tell process or thread that lock cannot be granted.*



Example (Avoiding Lost Updates)

Balance of account *anAcc* at t_0 is 75



© Wolfgang Emmerich, 1998/99

13



Deadlocks

- **2PL may lead to processes waiting for each other to release locks.**
- **These situations are called deadlocks.**
- **Deadlocks have to be detected by the lock manager.**
- **Deadlocks have to be resolved by aborting one or several of the processes involved.**
- **This requires to undo all the actions that these processes have done.**

© Wolfgang Emmerich, 1998/99

14



Locking Granularity

- *2PL applicable to resources of any granularity.*
- *High degree of concurrency with small locking granularity.*
- *For small granules large number of locks required.*
- *May involve significant locking overhead.*
- *Trade-off between degree of concurrency and locking overhead.*
- *Hierarchical locking as a compromise.*

© Wolfgang Emmerich, 1998/99

15



Hierarchical Locking

- *Used with container resources, e.g.*
 - *file (containing records)*
 - *set or sequence (containing objects)*
- *Lock modes intention read (IR) and intention write (IW).*
- *Lock compatibility:*

	IR	R	IW	W
IR	+	+	+	-
R	+	+	-	-
IW	+	-	+	-
W	-	-	-	-

© Wolfgang Emmerich, 1998/99

16



Transparency of Locking

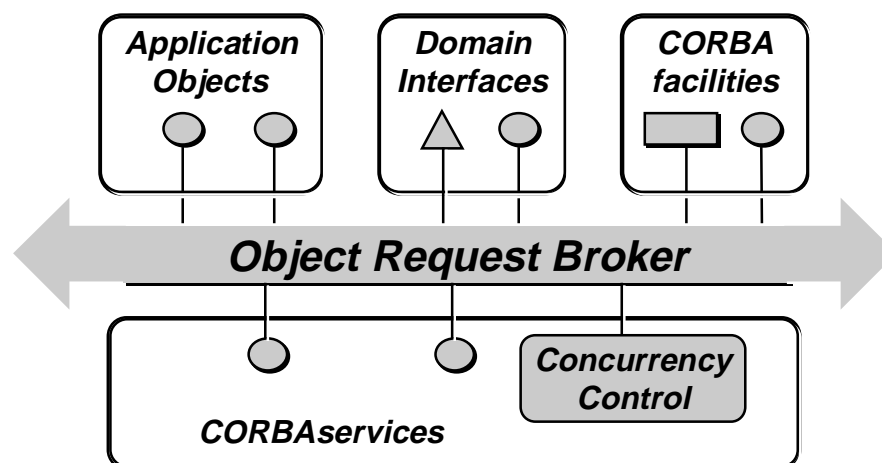
- **Who is acquiring locks?**
 - *Concurrency control infrastructure*
 - *Implementation of components*
 - *Clients of components*
- **First option desirable but not always possible:**
 - *Infrastructure must manage all resources*
 - *Infrastructure must know all resource accesses.*
- **Last option is undesirable and avoidable!**

© Wolfgang Emmerich, 1998/99

17



CORBA Concurrency Control Service



© Wolfgang Emmerich, 1998/99

18



Lock Compatibility Matrix

- **CORBA concurrency control service supports hierarchical locking.**
- **Upgrade locks for decreasing probability of deadlocks.**
- **Compatibility matrix:**

	IR	R	U	IW	W
IR	+	+	+	+	-
R	+	+	+	-	-
U	+	+	-	-	-
IW	+	-	-	+	-
W	-	-	-	-	-



Locksets

- **A lockset is associated to a resource (usually in the implementation of that resource).**
- **Each shared resource has a lockset.**
- **Operations of that resource acquire locks before they access or modify the resource.**



The IDL Interfaces

```
interface LocksetFactory {  
    LockSet create();  
};  
  
interface Lockset {  
    void lock(in lock_mode mode);  
    boolean try_lock(in lock_mode mode);  
    void unlock(in lock_mode mode);  
    void change_mode(in lock_mode held,  
                    in lock_mode new);  
};
```



Summary

- ***Lost Updates and Inconsistent Analysis***
- ***Locking and Lock Compatibility***
- ***Two-Phase Locking***
- ***Hierarchical Locking***
- ***Implicit vs. Explicit Locking***
- ***CORBA Concurrency Control Service***