The Transaction Concept: Virtues and Limitations

Jim Gray, 1981

Very Large Data Bases (VLDB) conference

V. Vikram Raj

October 12, 2005

About the author [http://research.microsoft.com/~Gray/]

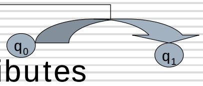
- □ Part of Microsoft's Research Group
- □ Interests include databases and transaction-processing systems
- □ Currently working on building supercomputers with commodity components
- □ Also working to build world-wide telescope
- □ ...

About the paper

- □ Very Large Data Base (VLDB) Endowment Inc. [http://www.vldb.org/] promotes and exchanges scholarly work in databases and related fields
- □ Paper presented at VLDB conference at Cannes in 1981
- □ 149 citations till date

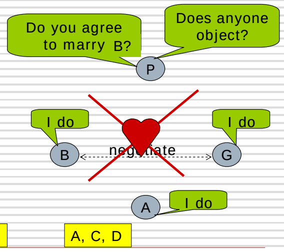
Motivation

- □ To understand the transaction mechanism as a tool to provide fault-tolerance to applications, which could be distributed
- □ To understand the methods of implementing the transaction mechanism
- □ To understand their limitations



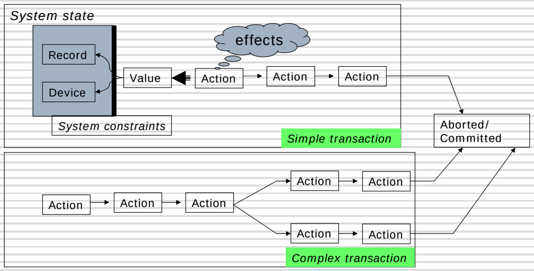
Transaction - attributes

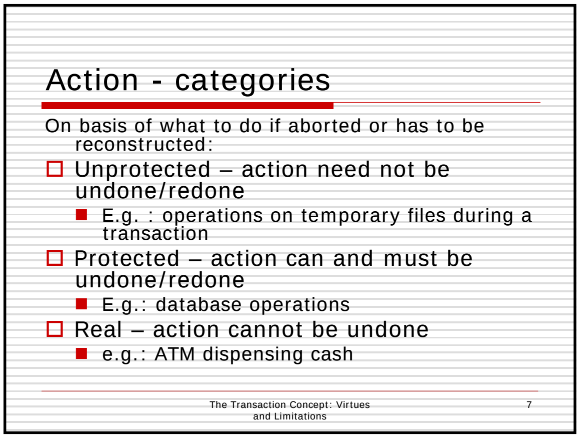
- □ A - Atomic
- □ C - Consistent
- □ I - Isolated
- □ D - Durable



In a country where gay marriages are prohibited, and 10 men and 10 women are to be married simultaneously, there should not be a case where there is one gay couple at the end.

Transaction - description

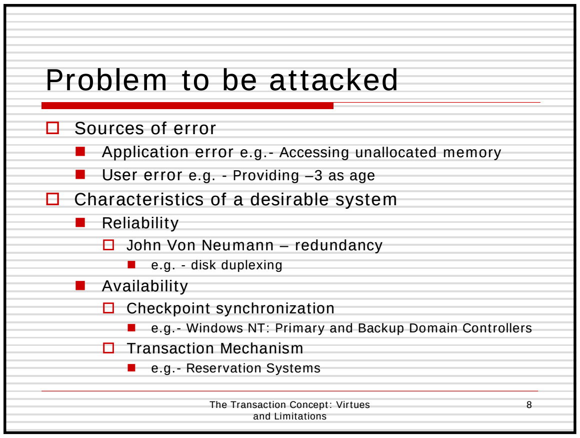




Action - categories

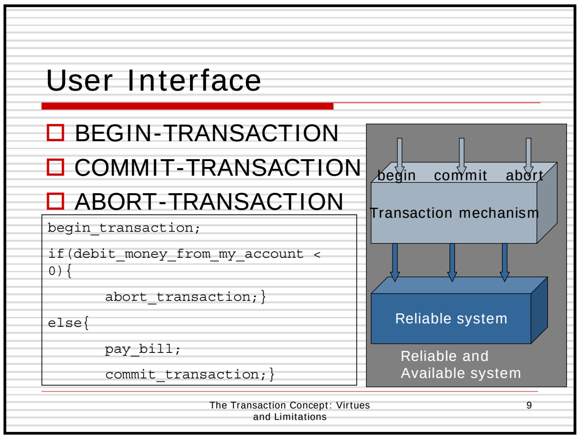
On basis of what to do if aborted or has to be reconstructed:

- □ Unprotected – action need not be undone/redone
  - E.g. : operations on temporary files during a transaction
- □ Protected – action can and must be undone/redone
  - E.g.: database operations
- □ Real – action cannot be undone
  - e.g.: ATM dispensing cash



Problem to be attacked

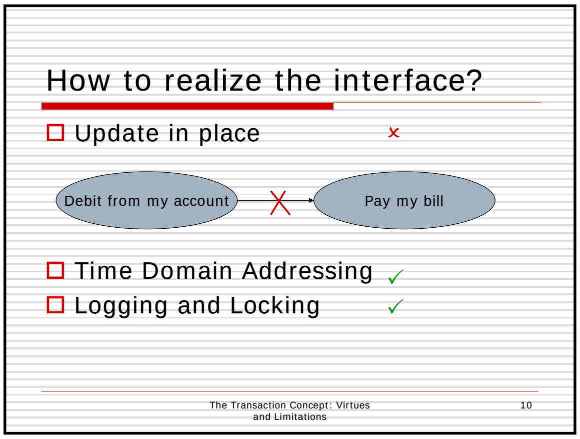
- □ Sources of error
  - Application error e.g.- Accessing unallocated memory
  - User error e.g. - Providing –3 as age
- □ Characteristics of a desirable system
  - Reliability
    - □ John Von Neumann – redundancy
      - e.g. - disk duplexing
  - Availability
    - □ Checkpoint synchronization
      - e.g.- Windows NT: Primary and Backup Domain Controllers
    - □ Transaction Mechanism
      - e.g.- Reservation Systems



User Interface

- □ BEGIN-TRANSACTION
- □ COMMIT-TRANSACTION
- □ ABORT-TRANSACTION

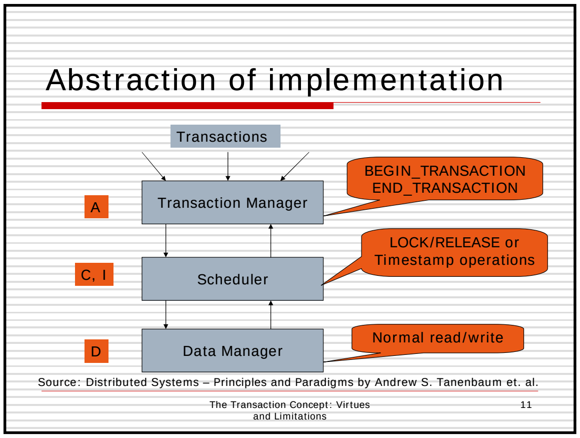
```
begin_transaction;
if(debit_money_from_my_account <
0){
        abort_transaction;}
else{
        pay_bill;
        commit_transaction;}
```



How to realize the interface?

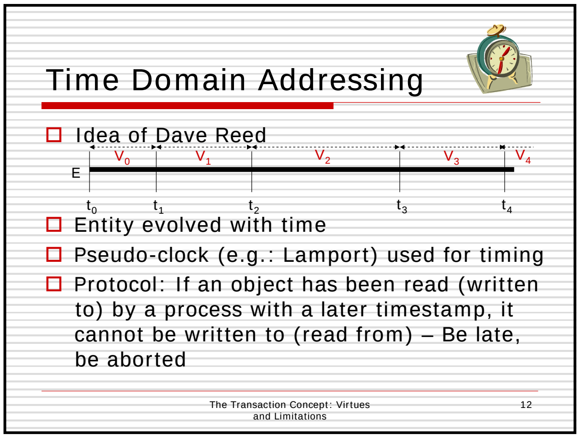
- □ Update in place ✗

- □ Time Domain Addressing ✓
- □ Logging and Locking ✓



Abstraction of implementation

Source: Distributed Systems – Principles and Paradigms by Andrew S. Tanenbaum et. al.



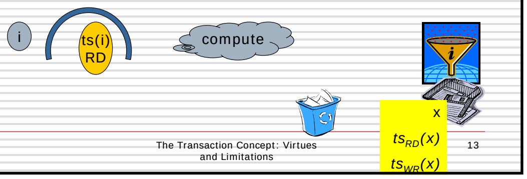
Time Domain Addressing

- □ Idea of Dave Reed

- □ Entity evolved with time
- □ Pseudo-clock (e.g.: Lamport) used for timing
- □ Protocol: If an object has been read (written to) by a process with a later timestamp, it cannot be written to (read from) – Be late, be aborted

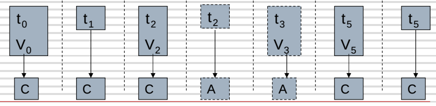
TDA – terminology

- Data item : x
- Read timestamp of x : $ts_{RD}(x)$
- Write timestamp of x : $ts_{WR}(x)$
- Timestamp of process i : $ts(i)$



TDA - Walkthrough

ts_{RD}	-	0	0	1	4	4	4
ts_{WR}	-	0	1	3	3	3	4
ts_i	0	1	2	2	3	5	5
op	WR	RD	WR	RD	WR	WR	RD



ACID checklist

- Atomicity – Ability to rollback using commit records
- Consistency – Loser is aborted; history not rewritten
- Isolation – Serializability through timestamps
- Durability – Reflecting upon commit

TDA – Pros and Cons

- ✓ Concurrency problem solved
- ✓ Reliability problem solved
- ✓ No deadlocks
- ✗ Reads are writes
- ✗ Waits are aborts; more transactions cause more aborts
- ✗ Timestamps force single granularity
- ✗ Real operations – pseudo time?

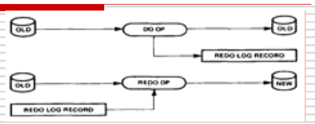
performance

Logging and Locking

- Used by Greeks – Ariadne & Theseus
- Each undoable (protected) action should create an undo (and redo) log along with the action which would allow the action to be undone (or redone)
- Exceptions
  - Unprotected – no log required
  - Real – defer action until commit

Logging - nuances

- Real actions

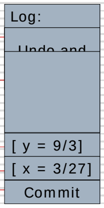


- Restartability
  - If operation is already undone/redone, the operation should not damage or change object state
  - Accomplished with version/sequence numbers
- Transaction committing to multiple logs
  - Speak up when given a chance

Logging - walkthrough

```
input x, y;
begin_transaction:
x = x + 2;
if (y == 0){
   abort_transaction;
  return -1;
 }
 y = y * 3;
 x = x / y;
commit_transaction;
```

x =25, y = 3

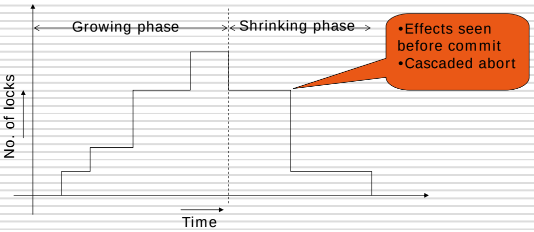


Locking nuances

- □ Concurrent transactions
  - T_1 and T_2 are concurrent. Output of T_1 → Input of T_2. T_1 aborts.- Cascading abort, confusion
  - Guess I/O sets and hold – not very successful
  - Lock object when accessed
- □ How to lock efficiently?
  - Predicate check – checking each predicate for members is expensive
  - Compromise – Fixed set of predicates as a directed acyclic graph

Locking protocol
Two phase locking

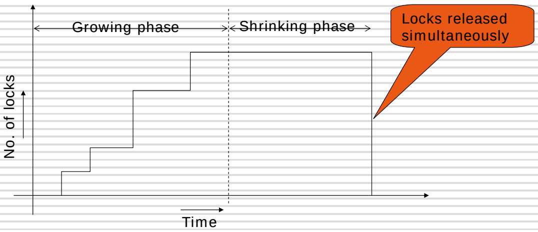
- □ What is the problem?



•Effects seen before commit
•Cascaded abort

Strict two-phase locking

- □Still, deadlocks could occur



Locks released simultaneously

ACID checklist

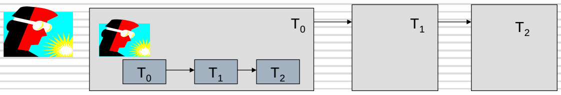
- □ Atomicity – Logs provide private workspace that allows rollback
- □ Consistency – Valid reader/writer holds the right locks
- □ Isolation – Serializability through locks
- □ Durability – New value reflects upon commit

Duality of approaches

- □ Logs tagged with version numbers for restartability vs time stamps
- □ TDAs archive old versions upon evolution ~ logs
- □ Locks used to update object header
- □ Every locking and logging trick has its time-domain counterpart –Dave Reed

Open problems- (1)Nested Transaction

- □ Views differ

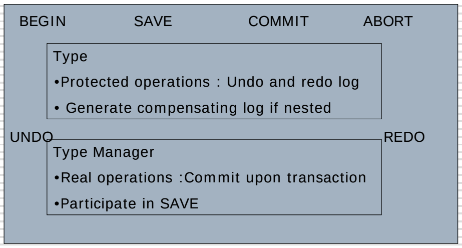


- □ Compensating Transactions
  - Transaction returns parameters to parent, which invokes if needed to be undone
  - Scratchpad – each state in database, loaded when active
- □ View: Is a nested transaction a transaction?
- □ Allows partial commit which might be desirable

Open problems- (2)Long-lived Transaction

- □ Transactions with lifetimes in days
  - Solution – 'Active' transactions hold locks
  - Updates of uncommitted transactions visible
- □ What if system restarted?
  - Transactions aborted – expensive
  - Salvaged with SAVE points.

Limitation : Integrating in the programming environment



BEGIN SAVE COMMIT ABORT

Type

- •Protected operations : Undo and redo log
- • Generate compensating log if nested

UNDO REDO

Type Manager

- •Real operations :Commit upon transaction
- •Participate in SAVE

Recent Developments

- □ In programming languages
  - Algis Rudys et. al. present a mechanism providing transactional rollback for 'codelets'. [http://www.cs.rice.edu/~arudys/papers/dsn2002.html]
- □ In operating systems
  - VINO – a transaction-based operating system [http://www.eecs.harvard.edu/~vino/vino/papers.html]
  - Uses software fault isolation (SFI) and transaction to minimize the damages of buggy and/or malicious "grafts."
- □ Effort to bring people working at different levels on transaction together - http://www.cs.wisc.edu/trans-memory/