08.05.2015

Transaction Processing

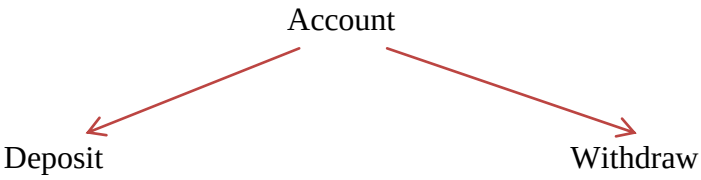
Chapter 16, 17

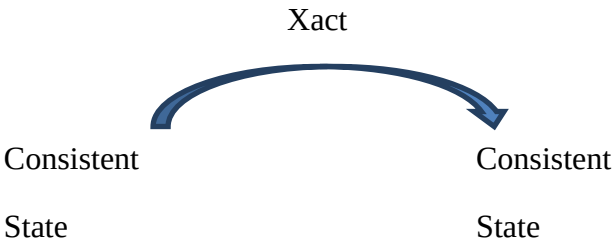
Transaction (Xact) is a small program that reads/writes to a database.

Jim Gray

Turing award

Online banking

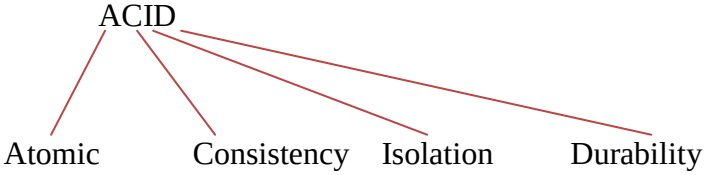




Power problem

Activity log

Transaction Properties



All valid transactions are atomic: a transaction starts and we always complete it.

Transactions are executed in such a way that the execution is identical with one of the possible serial execution.

T_1 T_2 T_3

T_2 T_1 T_3

T_3 T_2 T_1

…

1. T_1: Begin A=A+100, B=B-100, end
   T_2: Begin A= 1.06*A, B=1.06*B end

Commit: Completed

Abort: Interrupted

Write

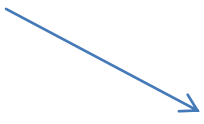
Read

Consider a possible interleaving (schedule)

2.

$T_{1:}$	A=A+100		B=B-100	
$T_{2:}$		A=1.06*A		B=1.06*B

$T_{1:}$	R(A),W(A)		R(B),W(B)	
$T_{2:}$		R(A),W(A)		R(B),W(B)



from Database point of view

1. A B
   1000 2000
   1100 1900
   1100*1.06 1900*1.06

2. A B
   1000 2000
   1100 1900
   1100*1.06 1900*1.06

Serializable Schedule

A schedule that is equivalent to some serial execution of transaction

Anomalies with Interleaved Execution

- Reading Uncommitted Data (WR conflicts "dirty read")

  C=Commit

$T_{1:}$	R(A),W(A)		R(B),W(B),Abort
$T_{2:}$		R(A),W(A),C	

- Unrepeatable Reads (RW Conflicts)

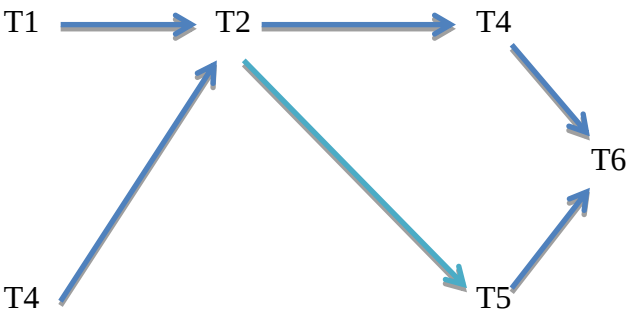
$T_{1:}$	R(A)		R(A),W(A),C
$T_{2:}$		R(A),W(A),C	

- Overwriting Uncommitted Data (WW Conflicts)

$T_{1:}$	W(A)		W(B),C
$T_{2:}$		W(A),W(B),C	

$W_1(A), W_2(A), W_2(B), C_2, W_1(B), C_1$

- convert it into a directed graph
- check if there is a cycle
- no cycle means no problem



T1, T3, T2, T4, T5, T6

T3, T1, T2, T5, T4, T6

- To prevent conflicts use locks shared lock
- If we just want to read it is ok if other people also read
- If a transaction wants to write a resource then in this case no other transaction can access the same resource exclusive lock

Deadlock: Two xacts are waiting a resource from each other