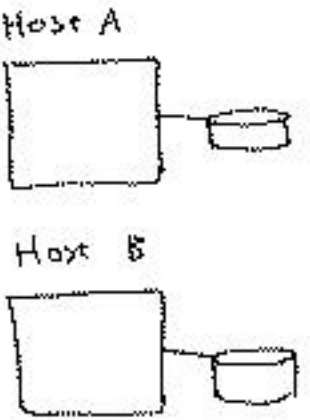
# Networked (Distributed) Databases

special considerations:

1. Each host (database) has its own transaction manager, with commit, recovery, etc.
2. Each host can fail & recover independently

EXAMPLE




T = A.withdraw(10);
B.deposit(10)

↓

actually:

T: Open Transaction

Open Sub Transaction @ Host A
lock X (write lock)
X = X - 10
commit sub transaction

Open Sub Transaction @ Host B
write lock (Y)
Y = Y + 10
commit sub Transaction.

Questions

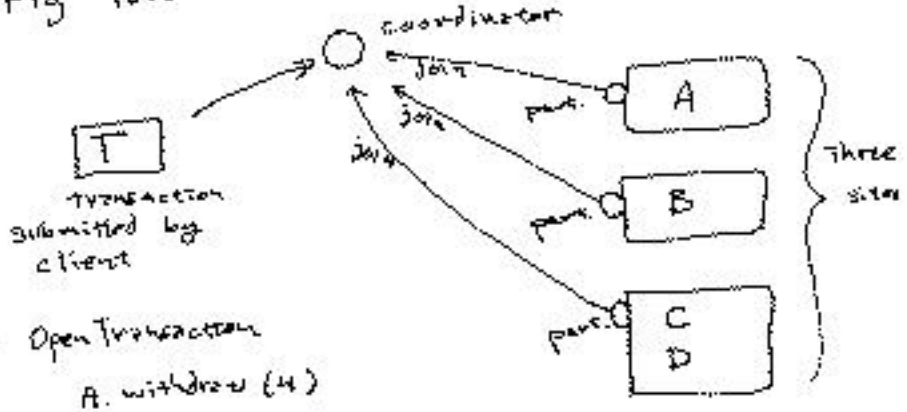
- what if Host B crashes after A commits its subtransaction?
- Under what conditions should subtransactions be run in parallel?

Better?

T: Open Transaction
Open Subtransaction @ Host A
Open Subtransaction @ Host B
write lock X @ A
write lock Y @ B
X = X - 10
Y = Y + 10
commit ??
↑
which order?
How to do this?

To solve these problems, we introduce, for each (distributed) transaction a coordinator (it can be a program on another host)

Fig 13.3




T: OpenTransaction
A. withdraw (4)
C. deposit (4)
B. withdraw (3)
D. deposit (3)
closeTransaction

T has unique ID

On each Host, we introduce "participant" (it is a proxy for T on that site)

A participant introduces a new state, used with TWO-PHASE COMMIT:

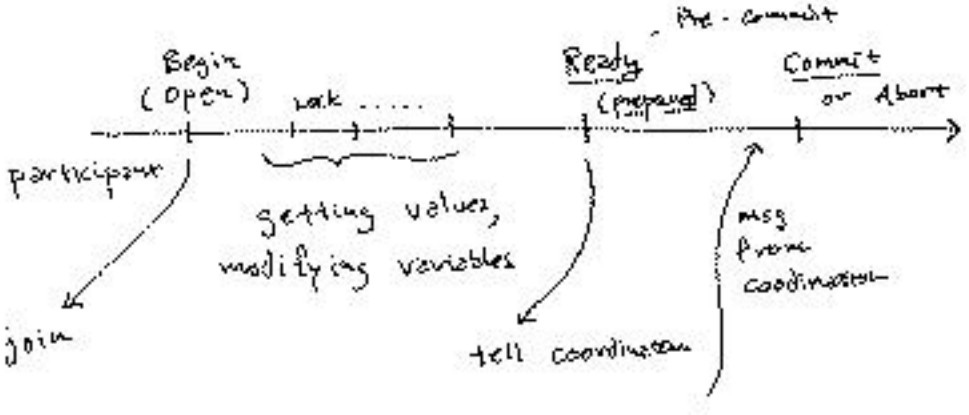
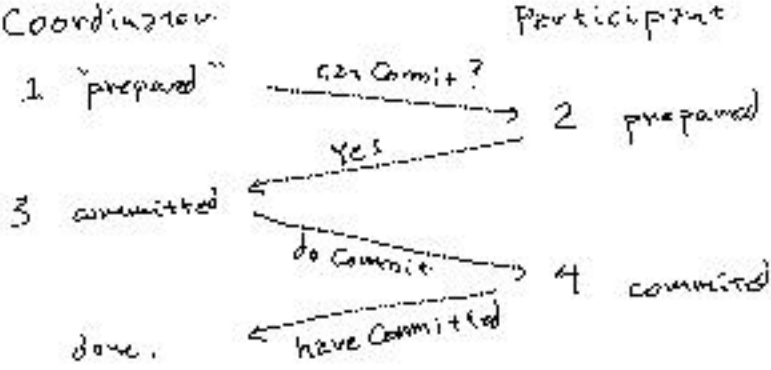


Fig 13.4 Operations for two-phase commit

| | |
|---|---|
| can Commit? | coord → part. |
| do Commit | coord → part |
| do Abort | coord → part. |
| have Committed | part → coord. |
| get Decision | part → coord. |




Note: do Commit is actually a vote of all participants

Note: do Commit technique only works if local transaction manager can guarantee that transaction is able to commit.

When is this true??