sync-xact

transactions

transaction: set of operations that occurs atomically

idea: something higher-level handles locking, etc.:

```
BeginTransaction();
int FromOldBalance = GetBalance(FromAccount);
int ToOldBalance = GetBalance(ToAccount);
SetBalance(FromAccount, FromOldBalance - 100);
SetBalance(ToAccount, ToOldBalance + 100);
EndTransaction();
```

idea: library/database/etc. makes “transaction” happens all at once

consistency / durability

“happens all at once” = could mean:

locking to make sure no other operations interfere (consistency)

making sure on crash, no partial transaction seen (durability)

(some systems provide both, some provide only one)

we’ll just talk about implementing consistency

transaction implementation ideas

(for consistency)

simple idea: do one transaction at a time

more efficient(?) idea:

- get list of operations
- lock everything operations use (in consistent order to prevent deadlock)
- do operations on list
- unlock everything

(also other strategies with different flexibility/efficiency)

Backup slides

implementing consistency: simple

simplest idea: only one run transaction at a time

implementing consistency: locking

everytime something read/written: acquire associated lock

on end transaction: release lock

if deadlock: undo everything, go back to BeginTransaction(), retry

- how to undo?
- one idea: keep list of writes instead of writing
- apply writes only at EndTransaction()

implementing consistency: locking

everytime something read/written: acquire associated lock

on end transaction: release lock

if deadlock: *undo everything*, go back to BeginTransaction(), retry

- how to undo?
- one idea: keep list of writes instead of writing
- apply writes only at EndTransaction()

implementing consistency: optimistic

on read: copy version # for value read

on write: record value to be written, but don't write yet

on end transaction:

- acquire locks on everything
- make sure values read haven't been changed since read

if they have changed, just retry transaction

implementing durability

what if there's a crash

we might have written some things but not others

most common approach: write-ahead logging

write list of intended operations + marker that list is complete

then do operations

after crash: check for intended operations

redo them *only if list is complete*