L16: Logging

Sam Madden
6.033 Spring 2014

Transaction terminology

Read(x) – get value of x from persistent store
Wtite(x,v) – write v to x in persistent store

```
xfer(a, b, amt):
    begin
      write(a,read(a) – amt)
      write(b,read(b) + amt)
      if read(a) < 0:
            print “Not enough funds”
            abort
      else:
            commit
```

All-or-nothing using shadow copy

```
xfer(bank, a, b, amt):
    copy(bank, tmp)
      tmp[a] = tmp[a] – amt
      tmp[b] = tmp[b] + amt
      rename(tmp, bank)
```

Shadow copy abort vs. commit

```
xfer(bank, a, b, amt):
    copy(bank, tmp)
      tmp[a] = tmp[a] – amt
      tmp[b] = tmp[b] + amt
      if tmp[a] < 0:
            print “Not enough funds”
            unlink(tmp)
      else:
            rename(tmp, bank)
```

Read with a log

```
read(log, var):
    commits = { }
      for record r in log[len(log)-1] .. log[0]:
            if (r.type == commit):
                commits = commits + r.tid
            if (r.type == update) and
               (r.tid in commits) and
               (r.var == var):
                return r.new_val
```

Read your own writes

```
read(log, var):
    commits = { }
      for record r in log[len(log)-1] .. log[0]:
            if (r.type == commit):
                  commits = commits + r.tid
            if (r.type == update) and
               (r.tid in commits or r.tid=cur_tid) and
               (r.var == var):
                  return r.new_val
```

Read / write with cell storage

```
read(var):
    return cell_read(var)

write(var, value):
      log.append(cur_tid, update,
                        var, read(var), value)
      cell_write(var, value)
```

Recovering cell storage from log

```
recover(log):
        done = { }
        for record r in log[len(log)-1] .. log[0]:
                if r.type == commit:
                        done = done + r.tid
                if r.type == update and r.tid not in done:
                        cell_write(r.var, r.old_val)    # undo
```

Cached read

```
read(var):
    if var not in cache:
      # may evict others from cache to cell store
      cache[var] = cell_read(var)
      return cache[var]
```

Cached read / write

```
read(var):
    if var not in cache:
      # may evict others from cache to cell store
      cache[var] = cell_read(var)
      return cache[var]

write(var, value):
      log.append(cur_tid, update,
                        var, read(var), value)
      cache[var] = value
```

Recovery for cached writes

```
recover(log):
    done = { }
    for record r in log[len(log)-1] .. log[0]:
        if r.type == commit:
            done = done + r.tid
        if r.type == update and r.tid not in done:
            cell_write(r.var, r.old_val)     # undo

    for record r in log[0] .. log[len(log)-1]:
        if r.type == update and r.tid in done:
            cell_write(r.var, r.new_val)   # redo
```

Abort

```
abort():
    for record r in log[len(log)-1] .. log[0]:
        if (r.tid == cur_tid)
            if r.type == update:
                write (r.var, r.old_val)  # undo
            if r.type == begin
                break
    log.append(cur_tid, abort)  # optional
```