# 6.033 Databses and RSM

Kevin Cho
R5 Peter Szolovits 1PM

Due April 20th 2016

## I Transactions

1. For the blue terminal, we already know what the table will looks like (the one given to use in the hands-on instructions). As for the red one, we have one more row added which is the username of chuck with a balance of $55 for Charles Robinson. This is because the transaction from the blue terminal has not been committed yet, the tables would not be the same.

| username | fullname | balance |
|---|---|---|
| bitdiddl | Ben Bitdiddle | 65 |
| alyssa | Alyssa P. Hacker | 79 |
| jones | Alice Jones | 72 |
| mike | Michael Dole | 83 |

Table 1: Blue Terminal

| username | fullname | balance |
|---|---|---|
| bitdiddl | Ben Bitdiddle | 65 |
| alyssa | Alyssa P. Hacker | 79 |
| jones | Alice Jones | 72 |
| mike | Michael Dole | 83 |
| chuck | Charles Robinson | 55 |

Table 2: Red Terminal

2. Chuck's account is not there because the red terminal is still not completed with the transaction yet. So, the updated list with Chuck's account is not updated from the inital commit from the blue terminal.

3. This time, the blue terminal will have all the accounts up to date. Thus means Chuck's account will also be there. This happens here and not in the 2nd question because both terminals have committed instead of just one.

4. Because the first transaction has not been committed, the second transaction will actually stop and block forever until aborted. Because the first transaction isn't finished yet, it prevents us from updating until it is finished.

5. Once we abort, the red terminal's transaction will be able to update successfully. If we commit, we get that mikes balance would be $73

6. Only after the commit are we able to see that the effects of the second terminal's transaction on the first one. This is because even though the updates are done, without a commit, the transactions are technically not added to the database.

## II Replicated State Machines

7. The balances are equal in the two tmp files.

8. The error message means there is a bug. The exact meaning is that the set of results from one transaction of the set of servers to another gives more than one transaction. The reason it shows this error is because there are two clients to the server. This creates a loop. Both servers give a result thus you have more than one pair which leads to a result.

9. The way we solve this problem is by having the client messages go to the servers one by one and returning one response. coord.py

```
#!/usr/bin/python

import sys, socket

replicas = map(int, sys.argv[1:])

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(('', 0))
s.listen(4)
print "\nListening on port " + str(s.getsockname()[1]) + " ..."

while True:
  conn, addr = s.accept()
  msg = conn.recv(4096)
  replies = []

  # Below goes your code. You need to write between 5-10 lines of code.
  # If you are writing more, you are doing something wrong.
  # the variables replies and replicas should be used.

  date = strftime("#, %a, %d %b %Y %H:%M:%S +0000\n", gmtime())

  for replica in replicas:
    connection = socket.create_connection(('localhost', replica))
    connection.send("%s:%s" % (msg, time))

    response = connection.recv(4096)
    replies.append(response)

  print replies
  conn.send(replies[0])
  conn.close()
```

client.py

```
    #!/usr/bin/python
import random, time

def client(servers):
  a = random.randint(0, 9)
  b = random.randint(0, 9)
  amt = random.randint(0, 10)
  results = []
  for s in servers:
    results.append(s.xfer(a, b, amt))
  # make a set out of results, and see if it has one item:
  if len(set(results)) != 1:
    print 'bug: xfer(%s,%s,%d) =' % (a,b,amt), results

import socket, sys, signal
```

```
class srv(object):
  def __init__(self, host, port):
    self.host = host
    self.port = port

  def xfer(self, a, b, amt):
    time.sleep(random.random() * 0.05) # Add some network delay
    fd = socket.create_connection((self.host, self.port))
    fd.send("%s %s %d" % (str(a), str(b), amt))
    resp = fd.recv(4096)
    ra, rb = resp.split(' ')
    fd.close()
    return (int(ra), int(rb))

ports = sys.argv[1:]
servers = []
for p in ports:
  servers.append(srv('localhost', p))

keep_running = True
def sigint(a, b):
  global keep_running
  keep_running = False

signal.signal(signal.SIGINT, sigint)
signal.siginterrupt(signal.SIGINT, False)

while keep_running:
  client(servers)
```