Name : Kreena Shah
Sapid : 60004210243
Batch : Comps C’32
Subject : Advance Database Management System (ADBMS)

# Experiment No 6 - Two Phase Locking Protocol

Aim : To implement 2 phase locking protocol

WriteUp:

Name : Koreena Shah
Sapid : 60004210243
Batch : C'32
Subject : ADBMS
Date of Performance : 27/11/2023
Date of Submission : 27/11/2023

## Experiment 6

Aim : Implement 2 phase protocol

Theory :

- As databases handle multiple users making concurrent transactions there is a scope of conflicts & data inconsistencies
- Locking protocols act as an excellent tool for ensuring data integrity by regulating access to shared resources
- The 2 phase locking protocol ensures a systematic & controlled approach to manage lockings adhering to principles of isolation & atomicity
- It plays pivotal role in preventing conflicts between transactions & guaranteering a reliable & coherent state of database
- Each working site in 2ppl has its own log & no global log.

Working :

Phase 1

- The site at which the query is requested becomes controller

In the case
- The transaction is initiated & the controller $C_i$ logs the request in its local log
- The transaction request is sent to the transaction managers across various sites where the changes required
- The transaction managers at these sites, log the incoming request
- If is requested changes at individual sites can be made causing no discripencies, then TM logs <Ready T>, means it's ready for changes
- If changes not applicable, <No T>
- The response generated at each site is transferred back to controlle manager

Phase 2
- The controller is waiting for responses from all sites
- If all sites respond <Ready T>, then the changes proposed are committed
- If even a single site refers the changes are aborted
- If some does not respond within time limit. It is assumed <No T> is response & transaction aborted
- The controller then logs <commit T> if commit, <Abort T> if aborting transaction

Conclusion :
We can conclude execution of 2 phase management system ensures seamless execution & reliability of concurrent transaction

Code:
Server

```
import socket
def ServerSoc():
    host = "127.0.0.1"
    port = 8000
    print("Server is running!")
    msg = "PREPARE"
    log = msg
    over = 0
    s_soc = socket.socket()
    s_soc.bind((host,port))
```

```
    s_soc.listen(2)
    while(1):
        replies = []
        print(f"Coordinator : {msg.upper()}")
        for i in range(3):
            conn,add = s_soc.accept()
            conn.send(msg.encode())
            data = conn.recv(1024).decode()
            replies.append(data.upper())
            print(f"Subordinator {i} {add} : {data.upper()}")
        if over == 1:
            break
        if ("ABORT" in replies) or (len(replies)<3) :
            print(f'at the abort stage the replies are {replies},and the length is
{len(replies)}')
            msg = "ABORT"
            print("TRANSACTION ABORTED!\nThe final log is: ", log+" "+msg)
            over = 1
        elif "SUCCESS" in replies:
            msg = "COMPLETE"
            print("TRANSACTION COMPLETED!\nThe final log is: ", log+" "+msg)
            over = 1
        else:
            msg = "COMMIT"
        log += " "+msg
ServerSoc()
```

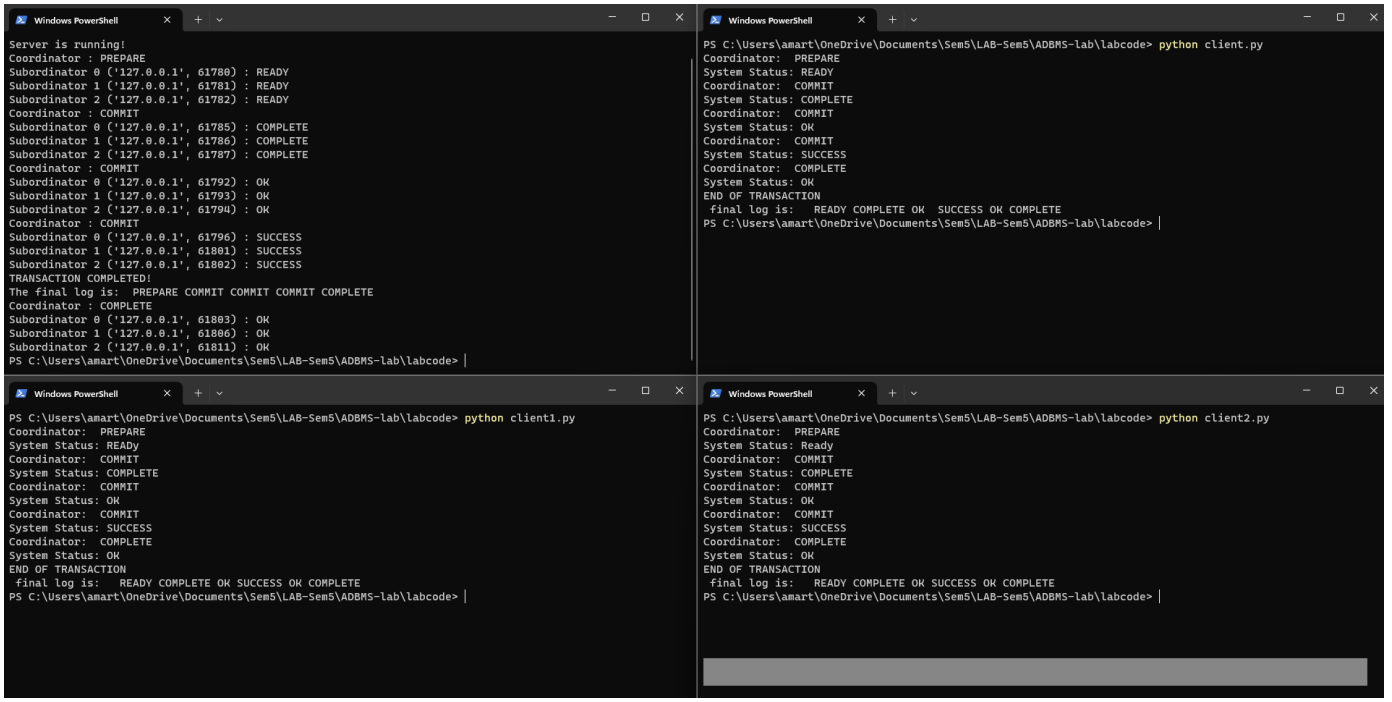
Client –

```
import socket
def ClientSoc():
    host = "127.0.0.1"
    port = 8000
    log = ""
    over = 0
    while(1):
        try:
            s_soc = socket.socket()
            s_soc.connect((host,port))
            rec_data = s_soc.recv(1024).decode()
            print("Coordinator: ", rec_data.upper())
            if rec_data.upper() == "ABORT":
                msg = "OK"
                print("TRANSACTION ABORTED!")
                over = 1
            elif rec_data.upper() == "SUCCESS":
                msg = "OK"
                print("TRANSACTION COMPLETED!")
                over = 1
```

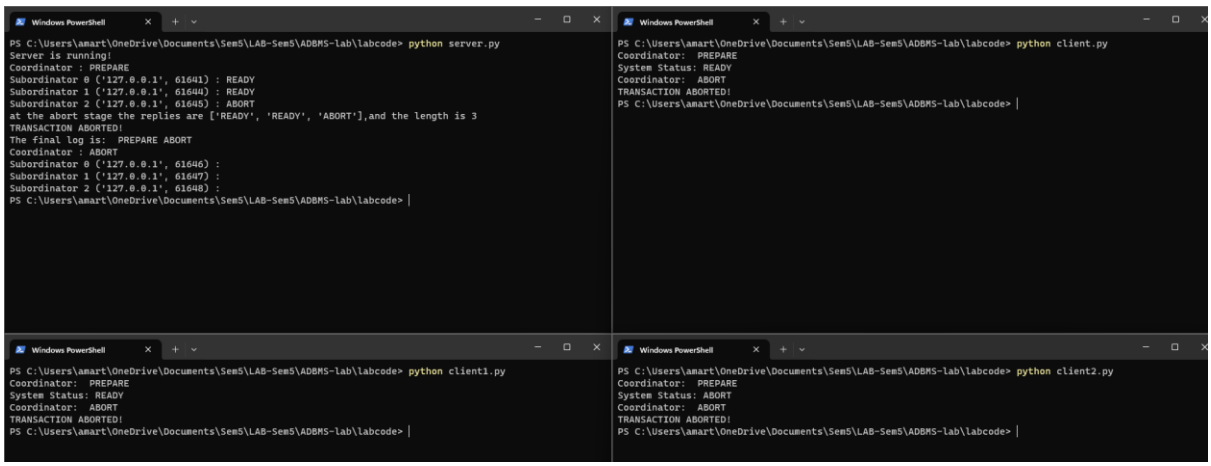
```
            else:
                msg = input("System Status: ").upper()
                log += " "+msg
                s_soc.send(msg.encode())
            if over == 1:
                break
            s_soc.close()
        except:
            print("END OF TRANSACTION\n final log is: ",log+" "+rec_data.upper())
            break
ClientSoc()
```

Output:

All in the ready phase:




One of the subordinators aborts:




Conclusion:

We conclude execution of 2 phase protocol management system ensures seamless execution and reliability of concurrent transaction.