CEL72, DSL, Monsoon 2020

Lab 6: Client Server based program to check data consistency.

Batch A, Student ID:

**Objective**

Implement Client Server based program to check data consistency.

Refer following link and implement objective as per suitable environment on your machine.

https://docs.oracle.com/cd/A57673\_01/DOC/server/doc/SD173/ch1.htm

**Consistency**

Consistency is the agreement between multiple nodes in a distributed system to achieve a certain value.

Specifically, it can be divided into strong consistency and weak consistency.

**Strong consistency:** The data in all nodes is the same at any time. At the same time, you should get the value of key1 in node A and the value of key1 in node B.

**Weak consistency:** There is no guarantee that all nodes have the same data at any time, and there are many different implementations. The most widely achieved is the ultimate consistency. The so-called final consistency means that the same data on any node is the same at any time, but as time passes, the same data on different nodes always changes in the direction of convergence. It can also be simply understood that after a period of time, the data between nodes will eventually reach a consistent state.

**CAP Theorem**

In theoretical computer science, the CAP theorem states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:

**1. Consistency:** Every read receives the most recent write or an error

**2. Availability:** Every request receives a (non-error) response, without the guarantee that it contains the most recent write

**3. Partition Tolerance:** The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

When a network partition failure happens should we decide to

* Cancel the operation and thus decrease the availability but ensure consistency
* Proceed with the operation and thus provide availability but risk inconsistency

**Conceptual Architecture**

Diagram

Description automatically generated

Fig.1. Conceptual Structure of the System.

The system consists of 2 clients, located at geographically distant location, accessing the replicas of same database, located at geographically distant location, for faster access. Data maintained in both the replicas must be consistent with each other.

According to the **CAP theorem** for distributed datastores states that any system can only achieve at the most two objectives from Consistency, Availability and Partition Tolerance. Assuming that the above-mentioned system prioritizes **Consistency** over Availability.

**Source Files**

**Client Program:**

**import** **pandas** **as** **pd**

**import** **os**

**import** **time**

*## initial data*

**def** init\_data(data):

**with** open("orders1.csv","w") **as** f:

**for** line **in** data:

**print**(line,file=f)

**with** open("orders2.csv","w") **as** f:

**for** line **in** data:

**print**(line,file=f)

**print**("Initialisation Done!")

**def** read\_data\_consistent(client\_id):

**print**(client\_id," wants to read ... ")

filename = "orders1.csv"

**if** client\_id == 2:

filename = "orders2.csv"

**while** "lock" **in** os.listdir():

time.sleep(2)

**print**(client\_id," is Waiting for read")

**print**("Reading begins ... ")

data = pd.read\_csv(filename)

**print**("Done !")

**return** data

**def** write\_data\_consistent(client\_id,data):

filename = "orders1.csv"

**print**(client\_id," is now writing ... ")

time.sleep(2)

**if** client\_id == 2:

filename = "orders2.csv"

**while** "lock" **in** os.listdir():

time.sleep(2)

**print**(client\_id," is Waiting for write")

f = open("lock","w")

**print**(client\_id," acquired lock")

**with** open("orders2.csv","a") **as** f:

f.write("**\n**"+data)

f.close()

**with** open("orders1.csv","a") **as** f:

f.write("**\n**"+data)

f.close()

time.sleep(2)

os.remove("lock")

**print**(client\_id," releases lock")

**print**(client\_id," is Done")

**print**(client\_id, " wrote ",data)

**if** \_\_name\_\_ == '\_\_main\_\_':

**print**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

write\_data\_consistent(1,"1,2,3,4")

**print**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

data = read\_data\_consistent(1)

**print**("================== 1 DATA ==================")

**print**(data)

data = read\_data\_consistent(2)

**print**("================== 2 DATA ==================")

**print**(data)

**print**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

write\_data\_consistent(1,"4,5,6,7")

**print**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

data = read\_data\_consistent(1)

**print**("================== 1 DATA ==================")

**print**(data)

data = read\_data\_consistent(2)

**print**("================== 2 DATA ==================")

**print**(data)

time.sleep(10)

data = read\_data\_consistent(1)

**print**("================== 1 DATA ==================")

**print**(data)

**Working of the System**

The system considers a scenario where 2 clients are accessing data from database copies which are geographically nearer to them. The database is constantly edited by the clients as well as other processes or interacting entities. While working with this database copes it may enter into an inconsistent state (not agreeing with each other) if the operations aren’t performed in the order they were intended to. In order to maintain the copies in a consistent state following technique is used.

Before operating on a database copy (reading or writing) the client first checks whether the databases is locked or not. If the database is locked the operation is deferred until a later time. This ensured by creating a lock on the database before any operation.

**Before Reading:**

1.The client checks whether the database is locked (is being edited by some other process) If it islocked the process has to wait till the lock is released.

2. After the lock is released the data is read from the database.

**Before Writing:**

1. The client checks whether the database is locked (is being edited by some other process) If it islocked the process has to wait till the lock is released.

2. After the lock is released the process acquires the lock and starts it writing. After completing its operations, the lock is released.

\*Database locking is performed by creating a lockfile in a shared space. This is just done for demonstration purpose other techniques are used by actual database management systems.

**Test Procedures**

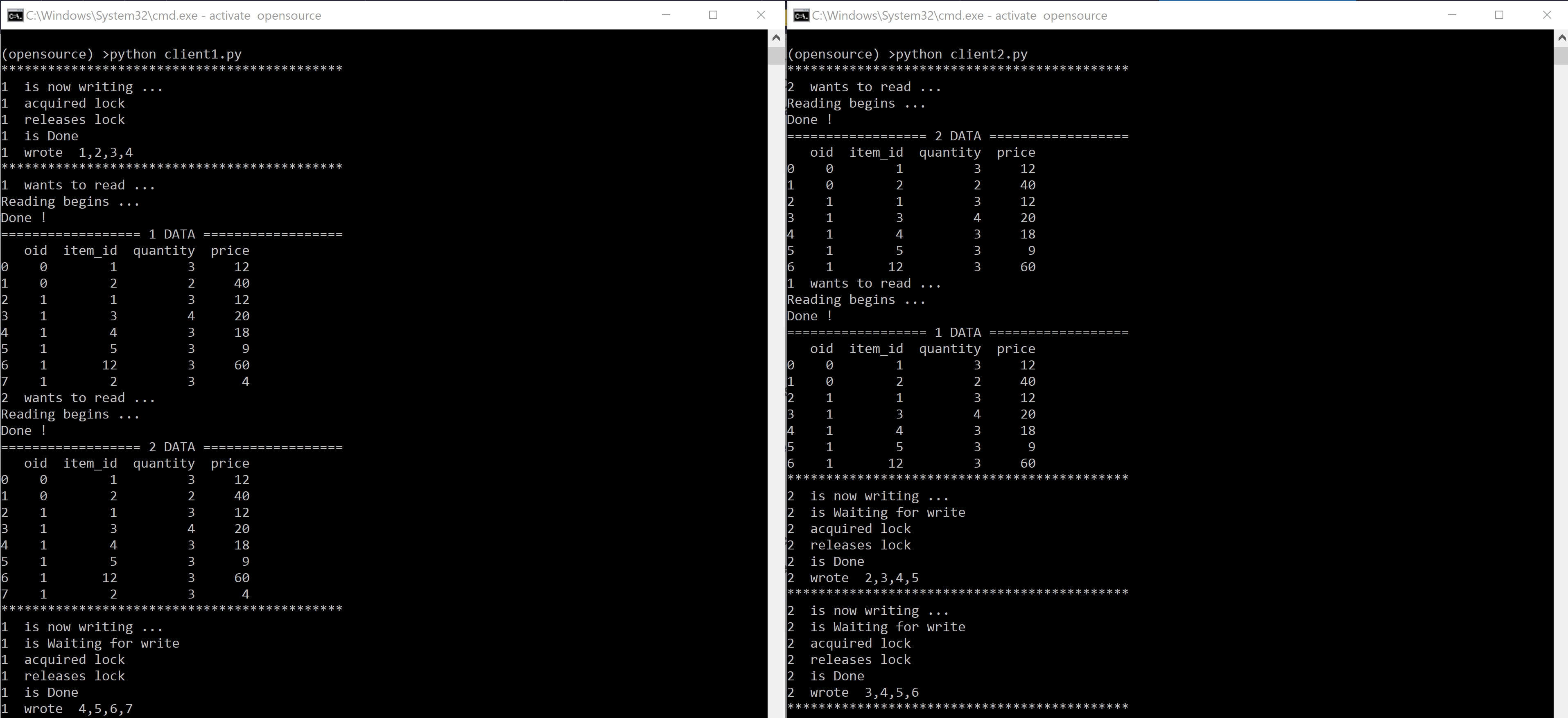


Fig.2. Two client programs simultaneously working and editing the database. Consistency is maintained using the lock file

The Demonstration covers 4 scenarios that may occur while accessing the database,

Client 1: Reading --- Client 2: Writing

Client 1: Writing --- Client 2: Reading

Client 1: Reading --- Client 2: Reading

Client 1: Writing --- Client 2: Writing

\*Demo gives a better idea of the overall working of the system.

**Conclusion**

Learned about the problem of data consistency in distributed systems and how various systems solve these problems Implemented one such simple solution for ensuring data consistency across a distributed database.

**Deliverables: (Place your work here directly under given steps)**

A solution description document (Times New Roman, 12pt, single spacing) containing:

1. Conceptual architecture of the distributed system you considered for assigned objective of experiment
2. Source files
3. Show process/event call steps for algorithm/method that you have implemented
4. Test Procedure for objective validation

**Evaluation**

Grading is performed for this assignment.

|  |  |
| --- | --- |
| **Election Algo laboratory work grading details Points 5 + Instructor Signature** | **Requirements** |
| 5 | * Explanation of source code and test cases ( Show at least two use cases of your work) (3M) * Documentation (2M) |

|  |  |
| --- | --- |
| **Feedback** | **Yes/No** |
| * Contents in this write up has been useful to perform experiment? * Level of understanding DS has improved? |  |

**References:**

1. *A. Tanenbaum and M. Steen, Distributed systems: principles and paradigms, Prentice Hall, Second Edition, 2005, ISBN: 0132392275.*
2. *Coulouris, G., Dollimore, J., Kindberg, T., and Blair G., Distributed Systems: Concepts and Design, Addison-Wesley,* [**Fifth Edition,** 2011, ISBN: 0132143011](http://www.cdk5.net/wp/).