```python
# Jamaree Moyer Database Design & Implementation
# package to handle thread-base parallelism
import threading
# package to handle time related systems
import time

print("Imported some packages")
```

```
Imported some packages
```

```python
# represents a distributed database system
class DatabaseNode:
  # initalization method
  def __init__(self, node_id):
    # unique identifier for each node
    self.node_id = node_id
    # data stored locally within the node
    self.data = {}
    # list of replica nodes
    self.replica_nodes = []

  # simulates a write operation on the batabase node
  def write_data(self, key, value):
    print(f"Node {self.node_id}: Write Operation- Key: {key}, Value: {value}")
    self.data[key]= value

    # iterates over each replica node to replicate the write operation
    for replica_node in self.replica_nodes:
      replica_node.receive_replication(key, value)

  # recieve replicated data from other nodes
  def receive_replication(self, key, value):
    print(f"Node {self.node_id}: Replication- Key: {key}, Value: {self.data.get(key,'Not Found')}")
    return self.data.get(key, None)

  # simulates a read operation on the database node
  def read_data(self,key):
    print(f"Node {self.node_id}: Read Operation - Key: {key}, Value: {self.data.get(key, 'Not found')}")
    return self.data.get(key, None)


print("Created the node that represents the ditributed database system")
```

```
Created the node that represents the ditributed database system
```

```python
# simulates a contious stream of write operation on a database node
def simulate_writes(node):
    # used to generate unnique keys for write operation
    i = 0
    # continous loop
    while True:
        node.write_data(f" k - {i}", f" v - {i}")
        # ensure unique key-value pair
        i += 1
        # pause execution for 2 secondes before next iteration
        time.sleep(2)

print("Define the methods to handle simulationg a continous stream of write operations")
```

```
Define the methods to handle simulationg a continous stream of write operations
```

```python
# create two node instances
node1 = DatabaseNode(1)
node2 = DatabaseNode(2)

# set up replication between the two nodes
node1.replica_nodes.append(node2)
node2.replica_nodes.append(node1)

print("initiialized the node instances and setup node replication")
```

```
initiialized the node instances and setup node replication
```

```python
# start writ operations for nodel in a separate thread
threading.Thread(target=simulate_writes, args=(node1,)).start()
```

```
Node 1: Write Operation- Key:  k - 0, Value:  v - 0
Node 2: Replication- Key:  k - 0, Value: Not Found
```

```
# initiates a read operation on nodel
node1.read_data("key0")
# pause 5 seconds to all write operations to be replicated between the nodes before reading again
time.sleep(5)
# performs a similr read operation on node2, allow for replication of write operations between the nodes
node2.read_data("key0")
```

```
Node 1: Read Operation - Key: key0, Value: Not found
Node 1: Write Operation- Key:  k - 1, Value:  v - 1
Node 2: Replication- Key:  k - 1, Value: Not Found
Node 1: Write Operation- Key:  k - 2, Value:  v - 2
Node 2: Replication- Key:  k - 2, Value: Not Found
Node 2: Read Operation - Key: key0, Value: Not found
```