Quiz2

Q1. Concerning the above figure, where does the MapReduce program store 1. Input data, 2. Intermediate files, and 3. Output data? (3 pts)

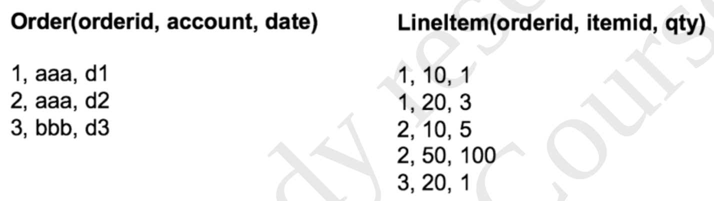
1. DFS/HDFS (1 pt)
2. Local FS (1 pt)
3. DFS/HDFS (1 pt)

Q2. Why is that when map workers fail, the tasks that are completed or in-progress at map workers are reset to idle and rescheduled but only in-progress tasks are reset to idle when reduce worker fails? (1pts)

When a **map task** completes, it sends the master the location and sizes of its intermediate files. If map worker fails, the intermediate result is no longer available to reducers. So both completed and in-progress task should be reset and rescheduled. (0.5 pts)

But **reducer** directly writes the results to file system. So only the in-progress task should be reset to idle when reducer fails. (0.5 pts)

Q3. Write the Map and Reduce tasks and their output for joining these two tables: (4pts)

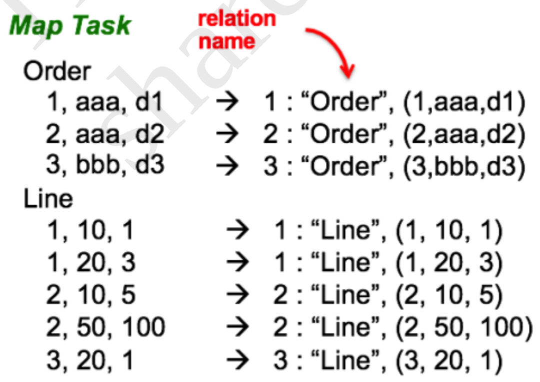


Order(orderid, account, date)	**LineItem(orderid, itemid, qty)**
1, aaa, d1	1, 10, 1
2, aaa, d2	1, 20, 3
3, bbb, d3	2, 10, 5
	2, 50, 100
	3, 20, 1

(pseudo code or plain words are both fine.)

Map task: (2 pts)

Use orderid as key, and table name together with other columns as value. Map each row in the table and emit the key-value pair.

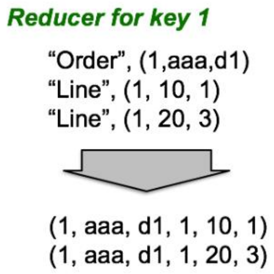


Map Task

relation name

```
Order
   1, aaa, d1      →  1 : "Order", (1,aaa,d1)
   2, aaa, d2      →  2 : "Order", (2,aaa,d2)
   3, bbb, d3      →  3 : "Order", (3,bbb,d3)
Line
   1, 10, 1         →  1 : "Line", (1, 10, 1)
   1, 20, 3         →  1 : "Line", (1, 20, 3)
   2, 10, 5         →  2 : "Line", (2, 10, 5)
   2, 50, 100       →  2 : "Line", (2, 50, 100)
   3, 20, 1         →  3 : "Line", (3, 20, 1)
```

Reduce task: (2 pts)
groups together all values (tuples) associated with each key and emit joined values.



Reducer for key 1

"Order", (1,aaa,d1)
"Line", (1, 10, 1)
"Line", (1, 20, 3)

(1, aaa, d1, 1, 10, 1)
(1, aaa, d1, 1, 20, 3)

Q4. Write a MapReduce program that multiplies two matrices A and B in **one stage/two stage.** You can assume that the matrices are provided to you in a file in a sparse matrix format. Each line of the file represents an element in a matrix. For example, a line: ['A', 0, 0, 1] indicates that A[0, 0] = 1. You may assume that both matrices are 5 x 5. (3pts)

One stage:

Map task:
For each element (i,j) of A, emit ((i,k), A[i,j]) for k in 1…5
Better: emit ((i,k), ('A', i, j, A[i,j])) for k in 1..5
Or just emit **((i,k), ('A', j, A[i,j]))** for k in 1..5
For each element(j,k) of B emit ((i,k), B[j,k]) for i in 1…5
Better: emit ((i,k), ('B', j, k, B[j,k])) for i in 1..5
Or just emit **((i,k), ('B', j, B[j,k]))** for i in 1..5
Reduce task:
emit **key = (i,k), value = Sumj (A[i,j] x B[j,k])**

Two stage:

1st Map Task:
For each matrix element A[i,j] : **emit(j , ('A', i, A[i,j]))**
For each matrix element B[j,k] : **emit(j , ('B', k, B[j,k]))**
1st Reduce Task:
For each key j, produce all possible products
For each value of (i,k) which comes from A and B,
i.e., ('A', i, A[i, j]) and ('B', k, B[j, k]): **emit ((i,k), (A[i, j] * B[j, k]))**

2nd Map Task:
The input would be the (key, value) from 1st Reduce task
Let the pair of **(((i,k), (A[i, j] * B[j, k]))** pass through
2nd Reduce Task:
For each (i,k), add up the values, **emit ((i,k), SUM(values))**