First question: How to recover the tuples between t=10 and t=12 if the node N failed at t=12 while the checkpoint was made at t=10?

There are two methods described in paper:

1) *"Input preservation: Each upstream primary node preserves output tuples in its output queues until they are safely stored at the downstream secondaries."*
2) *"Output preservation: The secondary is always "behind" the primary because its state corresponds to the last checkpointed state."*

Second question: What if two consecutive nodes failed?

This problem is not described in paper. And we also don't find solutions on website. But we think that it can be solved by recovering the two nodes step by step, i.e. first recover the upstream node and then use the tuples processed by the recovered upstream node to recover the downstream node. However, this may magnify the error. There must be some other methods dealing with the two-consecutive-node-failure situation, and we can look for that.

Third question: In precise recovery, how could the downstream node know that the upstream node fail, considering that the primary node N not works anymore?

The paper describes as follow (not in evaluation part, actually in method part):

"Passive standby provides repeating recovery for deterministic query networks. To make recovery precise, before sending any output tuples, the failover node must ask downstream neighbors for the identifiers of the last tuples they received and then discard all tuples preceding the ones identified. These requests can be made while the recovery node regenerates the failed state, achieving precise recovery without additional overhead.

For a non-deterministic query network, because the secondary may produce different duplicate output tuples when it takes over, the primary can only forward checkpointed tuples downstream. This constraint causes bursty output while also increasing the end-to- end latency."