Implementing Distributed Txns - 2PC.

Case: Zomato's 10 min Food Delivery:

Distributed Txn: A transaction that spans over multiple physical systems, m/c or Computers.

→ So as per the Case, Zomato guarantees the food delivery in 10 mins. To ensure this,

→ Zomato should accept order only when:
* Food is available in store
* Delivery Partner is available to deliver.

→ One Order
- has One Agent
- has One food item.

Functional Req1 → A user should not see "Order Placed" if cannot be fulfilled.

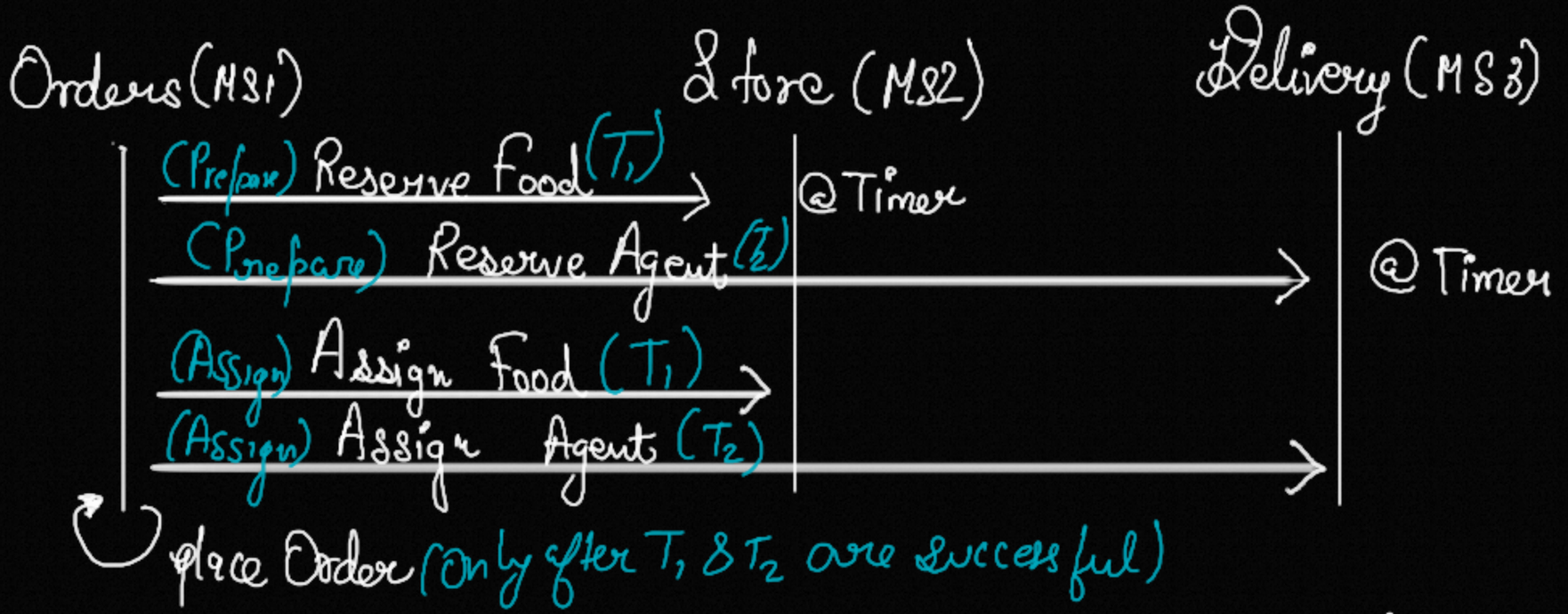
Functional Req2: → Resource Contention (Conflict over shared resource b/w several components) must be taken Care.

@

Implementing 2PC for this Case :

2PC : Split the entire flow in 2 phases : 1) Prepare — Reserve [Ensuring if whole 2PC Txn can happen automically]
2) Commit — Assign [After Ensuring Only we do the changes in db.]

Both Prepare and Commit Phase are done by a Service Known as Coordinator Service. (Here Order MS).

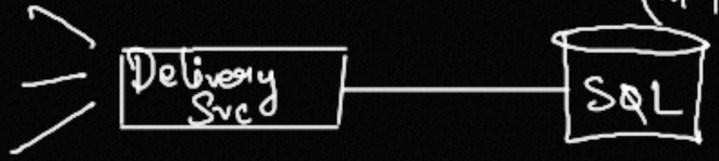


All these happen in 2 diff 2PC Txn (T_1 & T_2).
Also all these 4 txns are atomic individually too.
(Each 2PC Txn is automic in itself.)

→ If say T_1 gets failed at Assign phase, then the Timer of T_1 will release the resource locked by T_1 Prepare phase after some time.

③ DB Schema:

(A) Agents Table: (in Delivery Svc)



id	is_reserved	order_id
1	False/True	NULL

(Tells if agent is reserved for some Order Txn) (Currently Serving Order No)

If is_res = false and order_id ≠ NULL, it means the agent is not reserved, but is unavailable as he is out to delivery.

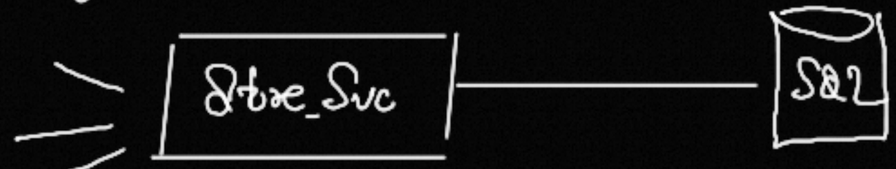
(B) Store Service:

Food Table

id	name
1	Burger

Packets Table:

did	food_id	is_reserved	order_id
100	1	False	NULL



Significance of Prepare Phase in 2PC Commit. How it narrows the blast radius? How it plays an important role in managing Concurrent txns on same resource and resource-contention?

The Prepare phase in 2PC, : Firstly finds the appropriate resource and locks it. (exclusive)

Secondly, Then it updates the is_reserved flag to 'true' on that resource. (which will obviously have prev. flag = false)

④ Now, in the first step of Prepare Phase, Concurrency is taken care as per db isolation levels and ACID properties of SQL DBs. After the first step of this phase, as the lock (Exclusive) has already been taken over row (resource), concurrency will not create issue. Also after the lock, resource-contention problem is also solved.
And when we mark is_reserved = True flag in the second step of the Prepare phase, we have removed any possibility of data inconsistency on this resource. (i.e ensured no data inconsistency.)
Thus we have reduced our blast radius for 2nd phase (Assign Phase) of this 2PC.
Thus 2nd Phase will likely to get succesfull if there is no network outrage or Service downtime. Also in that case we can try re-try mechanism as we are sure that above two are only possible reasons why Assign phase is not getting Successful. (Since both of these failure reasons will recover very soon),

I

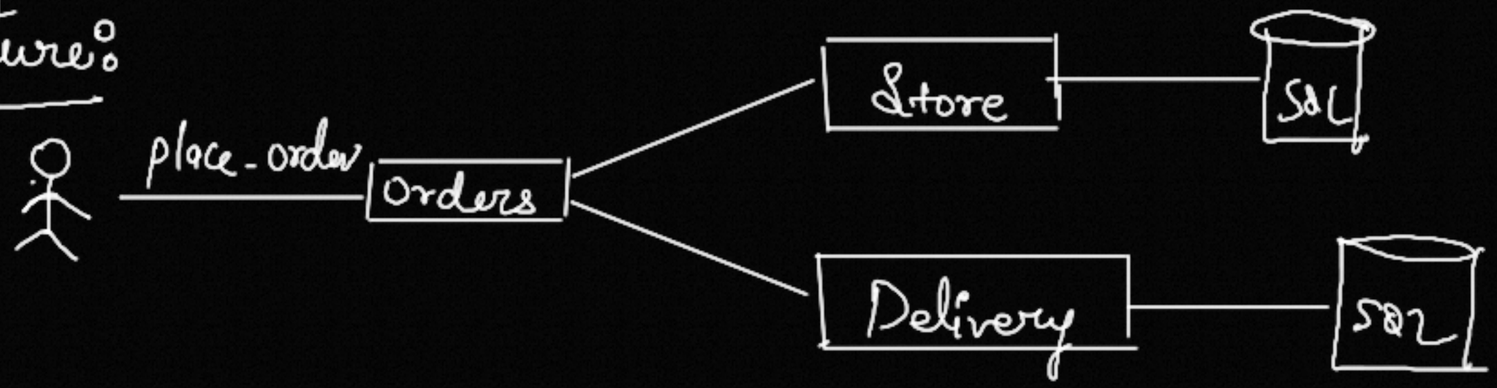
⑤ API Endpoints: (A) Delivery /delivery/agent/reserve → reserve a delivery agent (Prepare Phase)

/delivery/agent/book → assign delivery agent to an Order (Commit Phase)

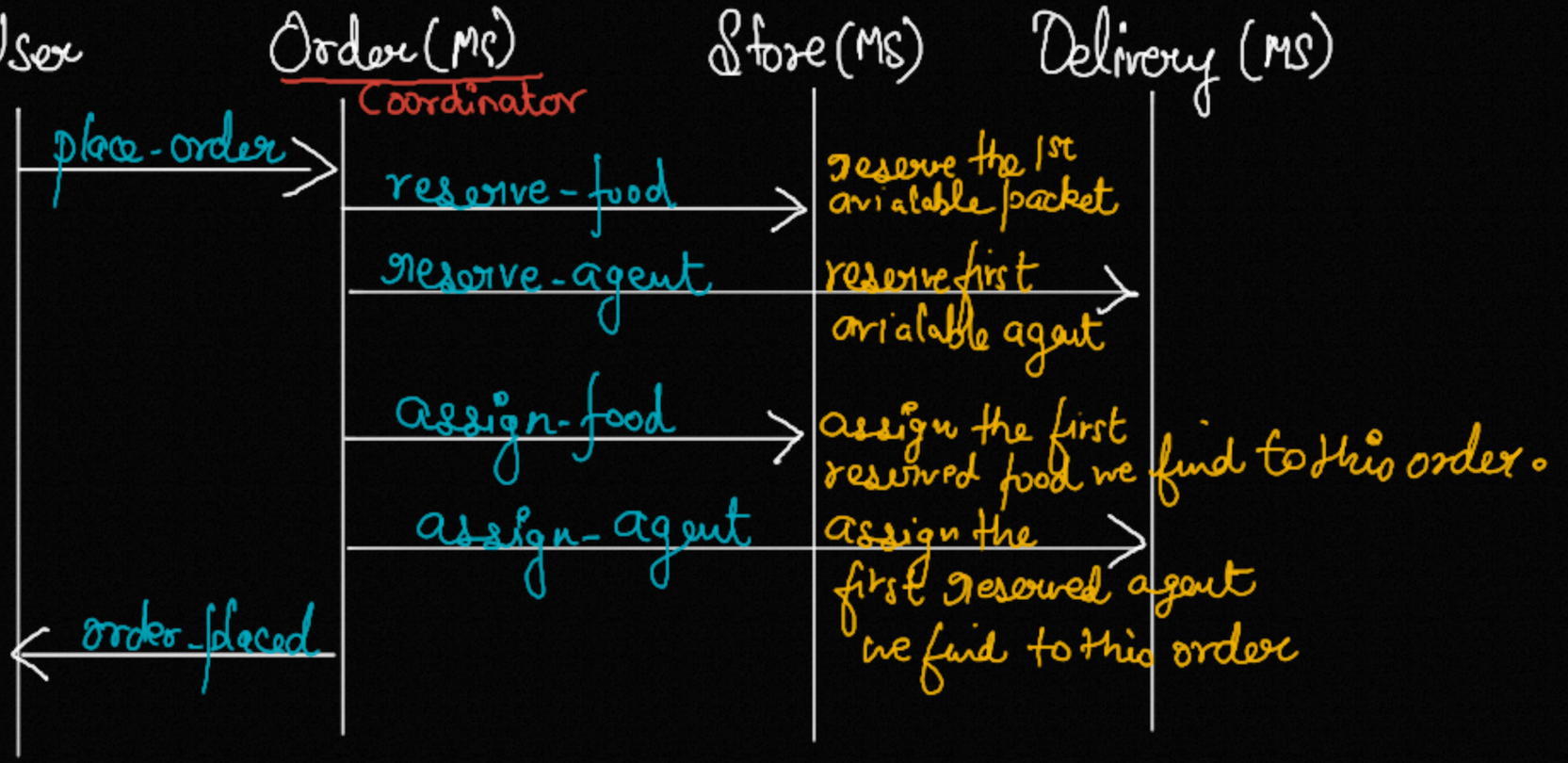
(B) Store: /store/food/reserve → reserve a food Packet (Prepare Phase)

/store/food/book → assign a food Packet to an Order (Commit Phase)

High Level Architecture:



(6)



Note: Since in the code, we have not implemented timer, once the food packet or delivery agent is reserved, they remain blocked in the system, even if their assign phase of 2PC failed. But as we implement this timer (coming), the problem will be solved.

Now as in the sequence diagram:

- Finding and reserving food happens atomically
- Finding and reserving agent happens automically
- Assigning of food Packet is done automically.
- Assigning of agent is done automically.

} All 4 core-updates are done individually atomic.

Concurrent Txns have Zero impact.

To place Order, all 4 should succed.

The reserve phase of 2PC ensures we have necessary resource quantity/capacity, thus reducing the burst radius of the further Txn to place that order.

→ If any service fails or any service encounters error to execute any phase of 2PC or any service, order service can also revoke the reservations.

[Now CODE ----]