AVIRUP DAS - MDS202013

DCBD - ENDSEM

07/07/2021

1) Let us consider the following two resources :-

i) Person ii) Vaccination Record

Here we could identify the vaccine receiver from a unique ID. Now when this person receives his first dose of vaccine, we create a new record for that patient using the POST command (non-idempotent method). When this person is about to receive the second dose, we need to check the type of vaccine, no. date of first dose, etc. so the older records we could use GET method (idempotent). Finally, we will update the second dose details using PUT (idempotent).

- Object model : Person, Vaccination Record
- Model URIs : /Person/{unique ID}
  /Person/{unique ID}/Record
  /Person/{unique ID}/Record/{first vaccine date}
  /Person/{unique ID}/Record/{second vaccine date}
  /Person/{unique ID}/Record/{Vaccine type}

We will alternate both the Person and Record details as JSON or XML.

To create record:
HTTP POST /Person/{unique ID}/Record/{Vaccine type}

To update record: HTTP PUT /Person/{unique ID}/Record/{first vaccine date}
HTTP PUT /Person/{unique ID}/Record/{second vaccine date}

To retrieve record: HTTP GET /Person/{unique ID}/Record/

7) i) Let us assume that the system already has two coordinator processes. Now if some process P sends a message to the coordinators, there will be a few cases:
i) No coordinators respond, ii) Both respond, iii) One of them responds.

If both coordinators respond then we are alone.
If one/or few respond then we choose new coordinators.

→ Process P will send an election message to all the processes which have some high priority scores.

→ P will wait for a time slot for it to receive response. Now if it does not receive any messages then it chooses itself as a coordinator and sends another message for the second coordinator, and wait for time slot and so on.

→ If P receives some messages, it will select those two processes as coordinators which have the highest priority score amongst the respondents.

→ If P receives one message, it will select that process as coordinator and send a second message. If there is no response then it elects itself.

Note that if at any stage P receives a message from process, then it will send a reconfirmation message and wait for time slot t, to for confirmation. If the confirmation is received, the processes are confirmed as coordinators, otherwise the whole process starts again.

7)(2) We will take a remote approach for $n (< 2)$ processes. First a message is sent to the coordinators of all of them respond then we are done, if atleast one of them do not respond then we have to elect that many coordinators. If none of them respond —

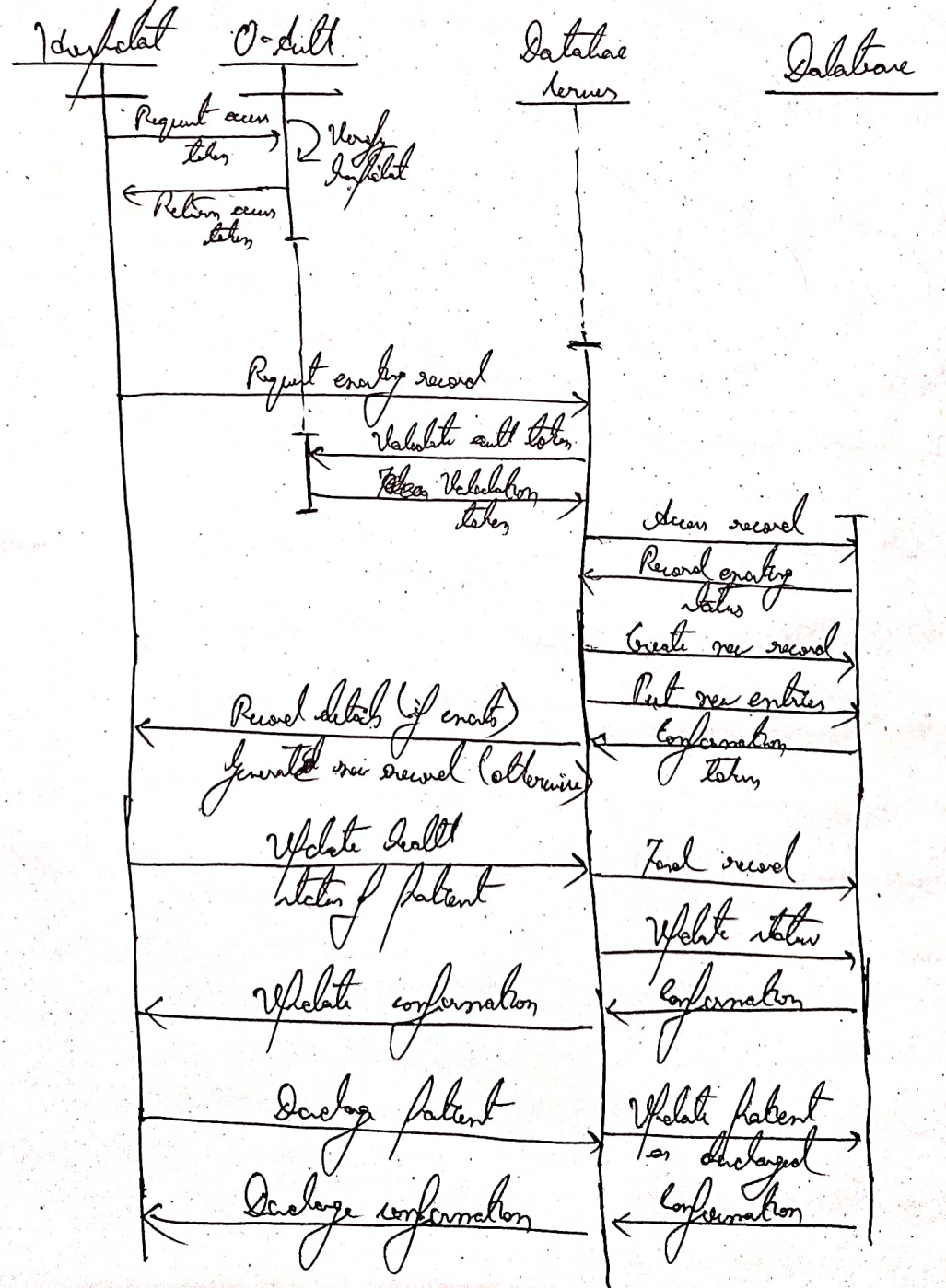
→ P sends a message to all processes for election and waits for time period t. If there is no response then it elect itself as one of the coordinators and moves forward with the election. If i responses are there (i = 1, 2, ... n−1) then it will elect the i-processes and itself.

→ If there are more than n responses, it will select those processes which have the highest priority score.

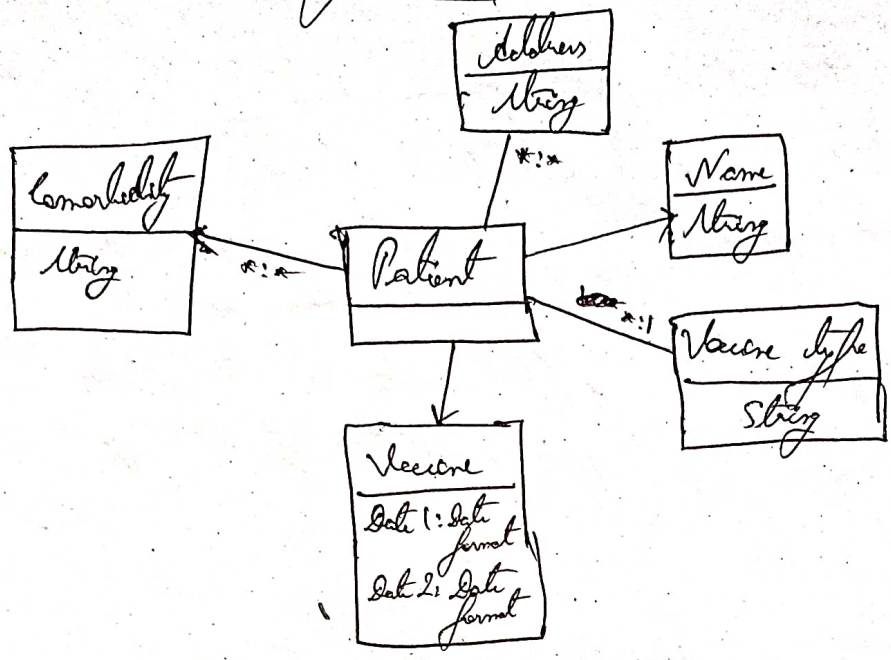
→ In this first case of message passing, the process P will wait for time t for the processes to respond otherwise it will elect itself and move on & will the election procedure for the rest of the coordinates remaining.

→ After the first case of message passing, process P will ask for a confirmation message from the 4 coordinates and wait for time t, for the coordinates to respond, if it receives confirmation messages within t, then it moves forward with the procedure otherwise we start again from the beginning.

3) First the hospital needs to send a message to the oAuth server so that oAuth server can authenticate the hospital as a legit account holder and pass on the request to the resource server. After verification, the hospital now has access to the resource server which has the database. Now the Thus hospital first tries to find the existing database or, if found then will be a request to update the same. If the database is non existent, the server will create a new one and will send the details back by the hospital.

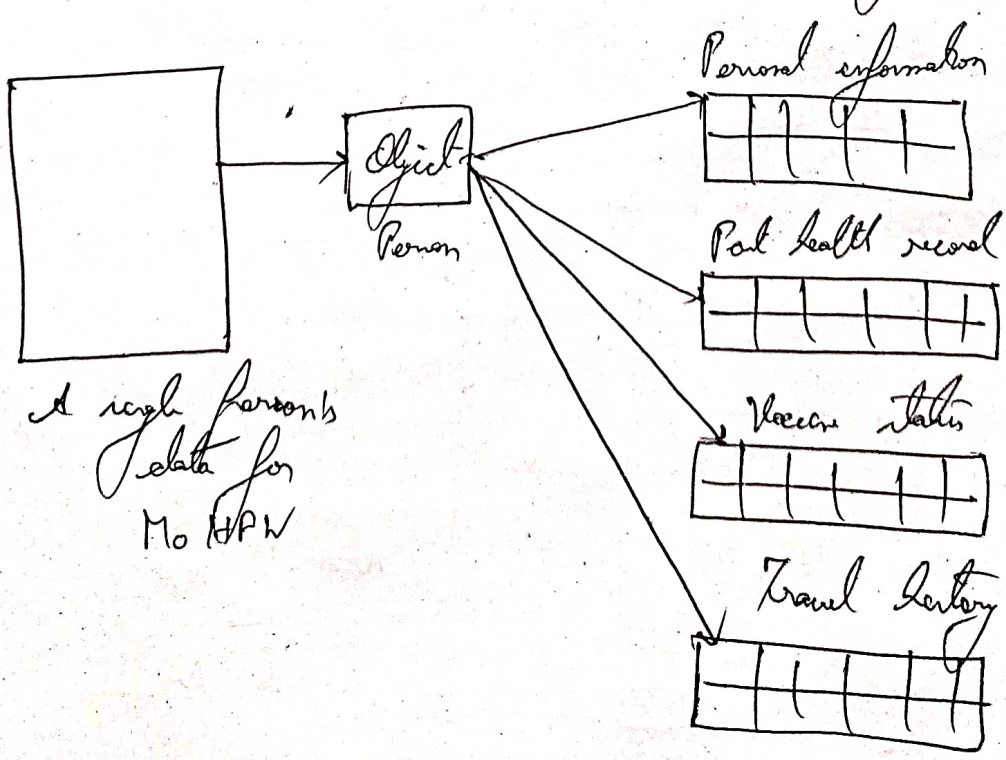


4) Partial class diagram :-



ORM :-

We have seen that it is much more intuitive and easy to represent data and in terms of object while being able to write simple queries. Clearly, there could be different types of object for the patient, for example - Personal information, Past health record, Vaccine status, Travel record. This can be represented using ORM.



1) (1) If N=4, M=2

1st round :-

All of the children will say no since they are not aware of the exact count. Although they will see the other children's heads, the exact count will be unknown since there is no info about their own status. So the a given child will either see 2 muddy heads or 1 muddy head depending on whether he himself is a muddy child.

2nd round :-

All of them conclude that atleast 2 of them are muddy. The two muddy children will see 1 muddy head and the clean headed children will see 2 muddy heads. Their clean headed won't be able to conclude about their own status, while the muddy ones will conclude that they have mud on their foreheads.

3rd round :-

From the second round they concluded that 2 children are muddy and they can identified themselves as muddy, this couldn't not have been possible if there were more. So from the previous round's answers, the clean children will identify themselves as clean in this round.

Round \ Child	1	2	3	4
1	NA	NA	NA	NA
2	Y	Y	NA	NA
3	Y	Y	N	N

NA → not sure

1)(2) If $N=4$ and M is a positive even number, M can be 2 or 4.

1)(1) → $M=2$ case has been discussed. Let us discuss it for $M=4$.

1st round → All of the children have mud on their foreheads and all see 3 other muddy children and no conclusion about themselves. They conclude that there are atleast 3 muddy children. But they can't pass in this manner, so there are 2, 3 or 4 muddy children.

2nd round → They concluded that there are atleast 3 muddy children but they still can see 3 muddy each and are not sure if they have mud on their own forehead. So, the ~~2nd round~~ ~~In the 2nd round~~ central inference would be there are atleast 3 muddy children.

3rd & round → In this round they know that 3 are muddy and atleast 3 should be muddy, so there is no strong conclusion.

4th round → In the final round everyone says yes as if exactly 3 were muddy then it could have been identified from the previous round - In the previous round if they would have seen 2 muddy and 1 non-muddy then they would have concluded in that round itself. So there are 4 muddy children and here all of them say yes.

Round \ child	1	2	3	4	
1	NA	NA	NA	NA	→ All muddy
2	NA	NA	NA	NA	
3	NA	NA	NA	NA	
4	4	4	4	4	

6) Note that when we are using vector clocks, it is difficult to determine the optimum number of bits that will be used to implement the integer values of the clock. If this number is very small, there might be a risk of overflow if there are millions of events. If the number is too large, there is an additional cost of storage and maintenance. Hence large clocks should is undesirable. Thus we can have to take a rough estimate of the no. of bits and continue the process. If we are in a situation when there is a risk of overflow, we can reset the vector clock for that particular process. Resetting the clocks helps us retain the functionality of the vector clock. We can even prevent clock overflow by carefully selecting the condition under which the clock is reset.

---

5) Let us take three persons.

Next we plot this matrix row-wise, meaning the row gives us the states of that person's branch. Now we plot this row into cells and pass through a mapper where the key is the person's name (row corresponding row) and the value is 1 if that cell has 1 or 0. These outputs are then shuffled and passed through a reducer where the values are reduced. Then we get the final output where the highest value has the highest no. of branch.

Input

Person	Branch visited		
	X	Y	Z
X	0	1	1
Y	0	0	1
Z	0	1	0

Shuffle

X	0	1	1
Y	0	0	1
Z	0	1	0

Reduce

X	2
Y	1
Z	1

→ X has the highest no. of branch