16/02/22

Tutorial-06

U19CS009

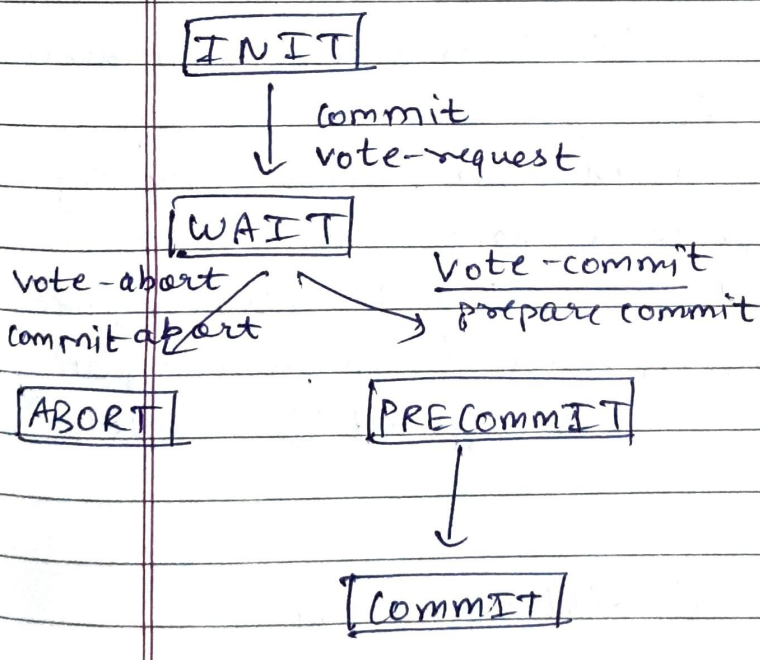
Page No.:
Date : / /

Q-1 Explain 3-phase commit protocol.

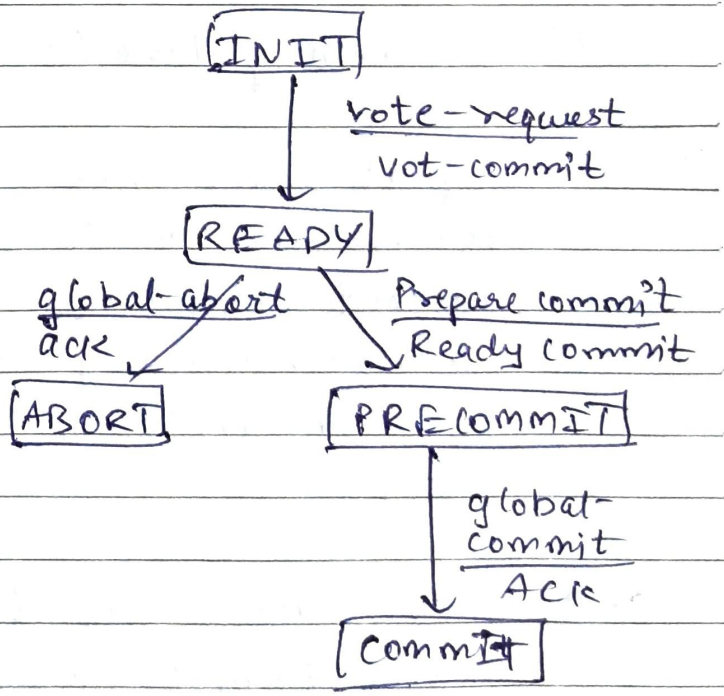
Ans → 3-phase-commit protocol (3PC) is an extension of two-phase-commit protocol (2PC) that avoids blocking problem under certain assumptions. Like 2PC, 3PC is also formulated in terms of a coordinator and a number of participants. The essence of the protocol is that the states of the coordinator and each participant satisfy the following:-

i) There is no single state from which it is possible to make a transition directly to either a (commit) or an (abort) state.

ii) There is no state in which ~~the~~ it is not possible to ~~make a transition directly to either a commit or a~~ make a final decision and form which a transition to a commit state can be made



Finit state machine for coordinator



finite state machine for participant

Page No.:
Date: / /

Q2 Explain your understanding for following.

a) Orphan messages → these types of messages whose receive events are recorded in the state of the destination process, but the send events are lost. Orphan messages are created when a process while sending message to another process is rolled back and restarted from the point of it's last checkpoint.

b) Domino effect → it occurs where rolling back one of the process causes one or more other processes to roll back.

c) Lost messages → messages whose 'send' event is recorded but 'receive' event remains unrecorded due to rollback of process.

d) Problem of Live Locks → Live locks occur when two or more processes continuously repeat same interaction with other process (requesting for resource etc) in response to changes in other process without doing any useful work. Due to this, the process status continuously changes, prevents them from completing task and makes it even more difficult to complete task.

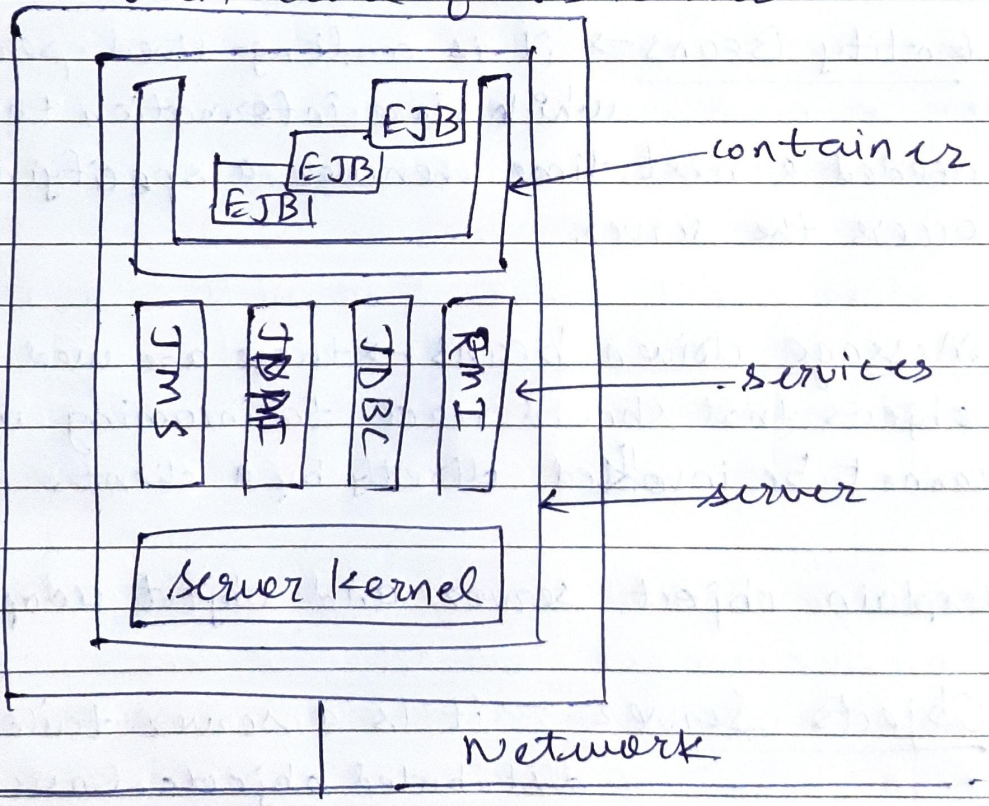
Q3 D/B remote object and distributed object

Ans → Remote object → some objects in distributed systems have their 'state' residing in a single machine. Only interfaces are made available on other machines. Such objects are known as remote objects.

→ Distributed Objects → objects contain encapsulated data called 'state' and operations on these data called as 'method'. Methods are available through interface. In Distributed system a strict separation b/w interfaces and objects at another machine. This organization is commonly refered as distributed objects.

Q-4 Explain Enterprise Java Beans (EJB)

Ans → An EJB is essentially a Java object that is hosted by a special server offering different ways for remote clients to invoke that object. Below is the diagram of general architecture of EJB server



Architecture of EJB Server

→ Typical services provided by EJB server include those for remote method invocation (RMI), database access (JDBC), naming (JNDI) and messaging (JMS).

Four kinds of EJB are :-

1) Stateless session beans → it is a transient object that is invoked once, does it work, after which it discards any information it needed to perform the service it offered to a client.

2) Stateful Session Beans → As contrasts to stateless session beans a stateful session bean maintains the client related state even after it does its works.

3) Entity Beans → it is a long-lived persistent object which store information that may be needed a next time whenever a specifying client accers the server.

4) Message driven beans → these are used to program objects that should react to incoming messages. They cannot be invoked directly by a client.

Q-5 Explain object server and object adapter.

Ans Objects Server → it is a server tailored to support distributed objects. Basically they by themselves does not provide a specific service. Specific services are implemented by objects residing in the server, so the object server provides the means to invoke the local objects based on request from remote clients. Object server only provides easy configuration of changing services

by simply adding and removing objects. Objects consists of two parts : (data) and (code) for executing methods. Whether or not these parts are separated or whether method implementation are shared by multiple objects depends on object server.

Object Adapter → Decision on how to invoke object & are commonly known as activation policies. These mechanism are needed to group objects as per policy. Such a mechanism is called as object adapter or object wrapper. An object adapter has one or more objects under it's control. Because a server should be capable of simultaneously supporting objects that require different activation policies, several object adapters reside in same server at same time. When an invocation request is delivered to the server, that request is first dispatched to the appropriate object adapter.

Q-6 Write different problems and different solutions with respect to

1). Server crashes → A request is carried out and a reply is sent which is a normal case. In second scenario a request is arrived and is carried out, but before sender can send reply, the server crashes. In third scenario a request has arrived, but before it gets carried out by the server, the server crashes. In these 2 cases, 2nd & 3rd are crashes. Following are solutions

a.) Wait until server responds/reboots and then try

the operation again. This way the client keeps try until server sends reply to it. This technique is called least one semantics.

b) Client give up immediately and reports back failure, known as at-most-once semantics.

c) It is guarantee nothing. When a server crashes, client gets no help and no promises about what happened.

2) Lost Reply Messages → lost replies can be difficult to deal with it. The obvious solution is rely on a timer sent by client's OS. If no reply comes, send the request once more.

→ Another way of solving is to try to structure all requests in an ~~independent way~~ idempotent way. Such operations which are necessary and can be done with no damage being done is said to be idempotent.

→ Another method is to have client each request a sequence number. This approach does require that the server maintains administration on each client. An additional safeguard is to have a bit in the message header that is used to distinguish initial requests from retransmissions.