Database Systems

Santhanalbhavathi.P
212720030076
IT B '22

Cat-III Subcode: IT16565

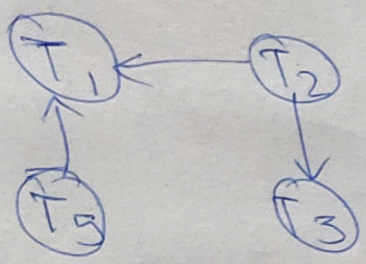
Part-C

8)
b)

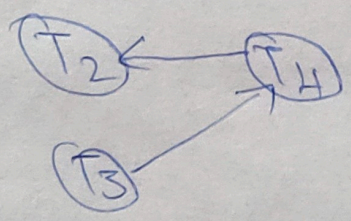
Deadlock handling

Deadlock prevention may result in unnecessary waiting and roll back. Furthermore, certain deadlock-prevention techniques may required more site to be involved in the ~~environment~~ execution of a transaction than would otherwise be the case

IF we allowed deadlocks to occur and relay on deadlock detection, the main problem in a distributed system is deciding how to maintain the wait for graph.



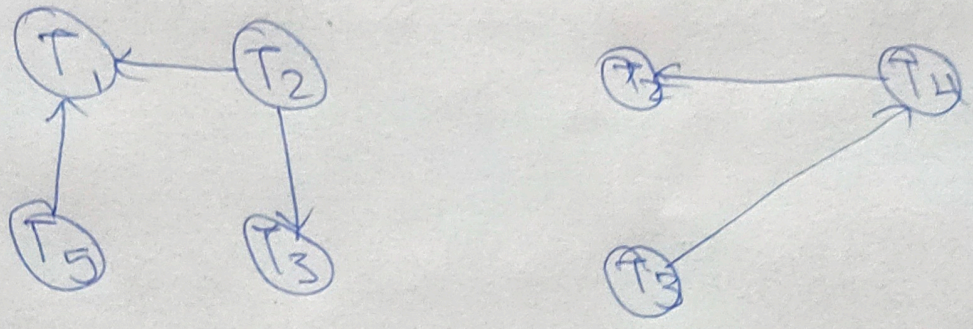
site S_1



site S_2

* Common techniques for dealing with this issue require that each site keep a local wait-for graph.

* The nodes of the graph correspond to all the transactions that are currently either holding or requesting any of the items local to that site



depicts a system consisting of two sites, each maintaining its local wait-for graph. T_2 and T_3 appear in both graphs, indicating that the transaction have requested items at both sites.

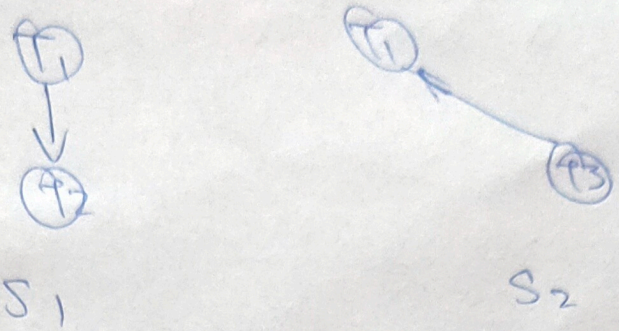
* These local wait-for graph are constructed in the usual manner for local transactions and data items

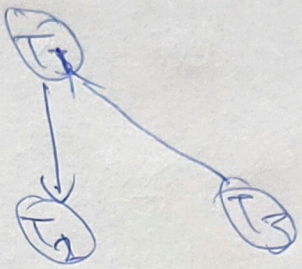
* When a transaction T_i on site S_1 needs a resource in site S_2, it send S_2, it sends a request message to site S_2. IF the resource is held by transaction T_j, the system inserts an edge $T_i \rightarrow T_j$ in the local wait for graph of site S_2.

* the centralized deadlock detection approach, the system constructs and maintains a global wait-for graph in a single site: The deadlock-detection coordinator.

* Since there is communication delay in the system, we must distinguish between two types of wait-for graphs.

* The real graph describes the real but unknown state of the system at any instance in time, as would be seen by an omniscient observer.





Coordinator

* Whenever a new edge is inserted in or removed from one of the local wait-for graph

* Periodically, when a number of changes have occurred in a local wait for graph

* whenever the coordinator needs to invoke the cycle-detection algorithm.

* when the coordinator invokes the deadlock detection algorithm, it search has been selected as victim.

This scheme may produce unnecessary rollbacks if:

False cycles:

* It exist in the global wait for graph. As an illustration, considered a snapshot of the system represented by the local wait-for graph. The T_2 releases the resource that it is holding in site S_1, resulting in the deletion of the edge $T_1 - T_2$ in S_1.

* A deadlock has indeed occured and a victim has been picked, while one of the transactions was aborted for reason unrelated to the deadlock.

* Both T_2 and T_3 are now rolled back, although only T_2 needed to be rolled back

* Deadlock detection can be done in a distributed manner, with several sites taking on parts of the task, instead of it being done at a single site.

* however, such algorithms are more complicated and more expensive.

Part-B

a)

b) Challenges in maintaining Data Consistency.

[redacted]

Data discrepancy occurs when the data in the target database deviates from the source database. The extent to which the data deviates depends on various factors, some of which may be intended and other unintended

Lift & shift workload to cloud.

Since the world is moving towards cloud, the lift & shift work load from on-Premises to cloud is the need of today's IT world.

Differences in source and target

Differences in source and target database configuration.

eg:- locales, endianness or database

Instantiation Errors:

Before migration or replication can begin, the target database(s) will need to be instantiated with the correct schema and constraints. Failure to do so will result in the source and target being out of sync.

Configuration errors

Improper and unintended configuration of replication nodes can cause discrepancies. This may present a task for database. No issues or work

Requirements for complex Data Consistency

* high speed, low impact data
* Support for heterogeneous database
* minimally intrusive
* zero downtime of source and target
* capability to identify data inconsistency
* low impact on hardware and network
* Data security
* Easy to use, understand, configure, deploy, and diagnose

7)

a) Fault - tolerant services

* key requirement: make a service fault tolerant
  eg:- lock manager, key-value storage system.
* state machines are a powerful approach to creating such services

state machine

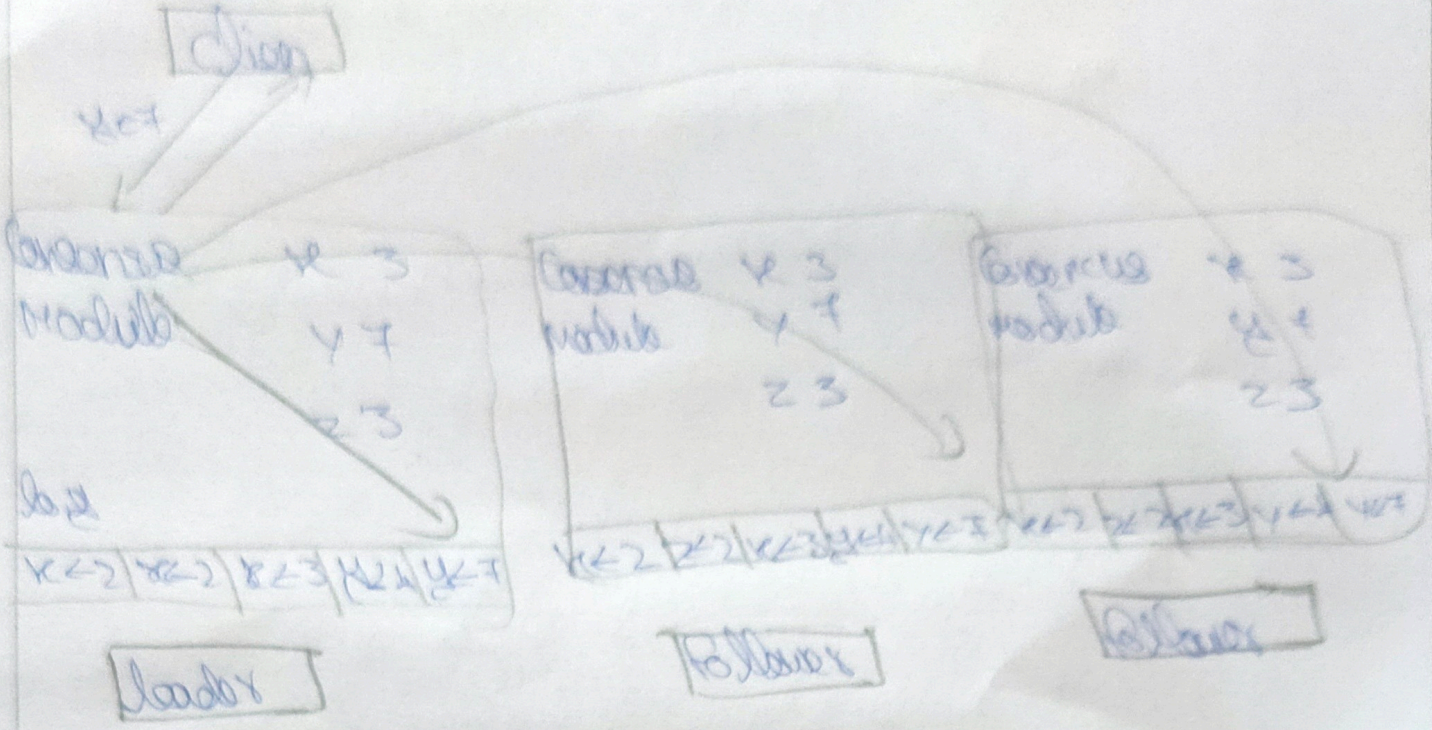
* Has a stored state, and receives inputs
* Makes state transitions on each input, and may output some results
  * Transitions and output must be deterministic

replicated state machine

* It has multiple nodes
* Replicated log state machine processes only committed inputs
  * Even if some of the nodes fail, state and output can be obtained from other nodes.

* Replicated state machine based on replicated log

* Example commands assign values to variables



leader declare log record committed after it is replicated at a majority of nodes. Update of state machine at each replica happens only after log record has been committed

* Replicated state machines can be used to implement wide variety of services

* Inputs can specify operations with parameters
* But operations must be deterministic.
* Result of operation can be sent from any replica
  * gets executed only when log record is committed in replicated log
  * usually sent from leader, which knows which part of log is committed

Eg:- Fault - tolerant lock manager
* state = lock table
* operations: lock requests and lock release
* output: grant, or roll back requests on dead lock

Eg:- Fault Tolerant key value store

State : key-value storage state

operation : Get() and Put() are First logged

* operations executed when the log record is in committed state

* Google spanner uses replicated state machine to implement key-value store

* Replicas of a partition form a Paxos group with one node as leader

* operations initiated at leader, and replicated to other nodes

Part-A

1

For the two disk mirrored case, we assume A disk and B disk. In order to lose data, A and B need to be failed at the same time. If A is already and with in 100,000 hours B disk B will fail, then data will be lost. The other case is B is already failed and within 100,000 hours A will fail and then data will be lost

For the first case, A disk is failed for 100 hours every 100,000 hours. So in order to make B to fail, it will need $100,000^2/100$ hours, because the other one, the time is reduced to $100,000^2/(2*100)$

2) Database indexing

Hash tables may also be used as disk-based data structures and database indices although B-trees are more popular in these applications. In multi-node database systems, hash tables are commonly used to distribute rows amongst nodes, reducing network traffic for hash joins.

3

Advantages	Disadvantages
Data retrival: Computer-based systems povide enhanced data retrieval techniques to retrieve data stored in files in easy and efficient way	**Data Redundancy:** It is possible that the same information may be duplicated in different files. this leads to data redundancy results in memory wastage
Editing: It is easy to edit any information stored in Computers in form of files	**Data inconsistency:** Because of data redundancy, it is possible that data may not be in consistent state

5 Map database management.

Systems are software programs designed to efficiently store and recall spatial information they are widely used in localization and navigation, especially in automotive applications.

It plays important role in the emerging areas of location-based services, active safety functions and advanced driver-assistance systems. Common to these functions is the requirement for an on board map database that contains information describing the road network.