IT 18305 Data base systems Semester - 3 Cat NO. 3	Mythra M B Tech IT-B

Part - A

1) Mean Time to failure - 100000 hours
Mean Time to repair - 10 hours.
Mean Time to data loss - $\frac{100000^2}{2 \times 100}$

3 Advantage:

1. Data Retrieval:

Computer-based systems provide enhanced data retrieval techniques to retrieve data stored in files in easy and efficient way

2. Editing.

It is easy to edit any information stored in computers ion form of files.

Specific application programs or editing softwar can be used for this purpose.

Disadvantages:

1. Data Redundancy:

It is possible that the same information may be ~~duplicated~~ duplicated in different files. This leads to data redundancy.

2. Data Inconsistency.

Because of data redundancy, it is

possible that data may not be inconsistent state.

5) ⇒ Map database Management systems:

⇒ software programs designed to efficiently store and recall spatial information.

⇒ widely used in localization and navigation especially in automotive applications.

⇒ When designed well, a map database enables the rapid indexing and lookup for a large amount of geographic data.

2) ~~B+ trees data be~~ 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.

4) Mysql enables retrictions to be placed on reuse of previous passwords. to establish password-reuse policy globally, use the password_history and password_reuse_interval system variables.

Part - B

6b. Data consistency - challenges

⇒ Data discrepancy occurs when the data in the target deviates from the source database. The extent to which the data deviates depends on various factors.

⇒ Using products that replicates data reliably, there remain potential causes of data discrepancy. If the goal of database is to be strictly consistent with the source, then it has to put processes and policy to ensure it;

CAUSES:

a) Migration effect:

⇒ Different kinds of migration tools are employed to facilitate the initial load of the target database from replication. Difference in configuration for handling data by the migration tools and replication products can result in data discrepencies.

b) Lift and shift workload to cloud:

⇒ since the world is moving towards cloud, the lift and shift of database workload from on-premises to cloud is the need today.

c) Replication Latency :

⇒ with asynchronous replication, there will be short lag between changes to the source database and delivery of those changes to the target.

⇒ Failure to meet the maximum latency requirement, however can potentially violate service level agreement levels or data compliance requirements.

d) User errors :

⇒ Target databases are often created to offload query processing from the source database. This enables operational reporting without impacting the application running on the source database.

Data consistency requirements:

⇒ High speed, low impact data comparisons.

⇒ Support for heterogeneous databases.

⇒ compatability for handling large data volumes.

⇒ Minimally intrusive.

⇒ Data security.

⇒ Easy to use, understand, configure, deploy and diagnose.

⇒ Data comparison reports for auditing purposes.

Part - C.

7a. Fault tolerant service using replicated state machines

⇒ Key requirements to mak a service fault tolerant.

Eg: lock manager, key-value storage system, ..

⇒ State machines are powerful approach to creating such services

⇒ A state machine

⇒ has a stored state and received inputs

⇒ Makes state transitions on each input and may output some results

- Transitions and output must be deterministic.

⇒ A replicated state machine is a state machine that replicates on multiple nodes.

⇒ All replicas must get exactly the same input.

- Replicated log: state machine processes only committed inputs

⇒ Even if some of the nodes fail, state and output can be obtained from other nodes.

Uses of Replicated state machine:

⇒ It can be used to implement wide variety of savices

⇒ Inputs can specify operations with parameters

⇒ But operations must be deterministic.

⇒ Result of operation can be sent from any replica.

- Gets executed when log record is committed in replicated log.

- Usually sent from leader, which knows which part of log is committed.

⇒ Example: fault tolerant lock manager

⇒ state: lock table.

⇒ operations: lock req., lock releases.

⇒ output: grant or rollback req. on deadlocks.

⇒ Centralized implementation is made fault tolerant by simply running it on a replicate state machine.

Fault tolerant key-value store

⇒ State: key-value storage state

⇒ operations: 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.

- Data is partitioned and each partition is replicated across multiple nodes.

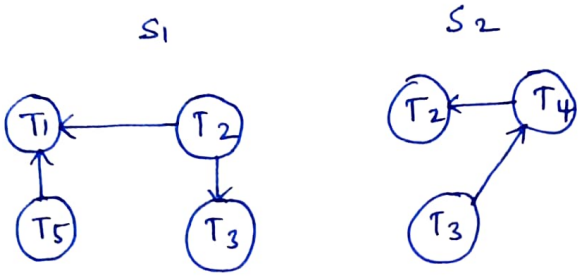
- Replicates of partition from a Paxos group with one node as leader

- operations initiated at leader and replicated to other nodes.

8b. **Deadlock handling:**

common techniques include that each site keeps 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.

For example, system consists of 2 sites, each maintaining its local wait-for graph.

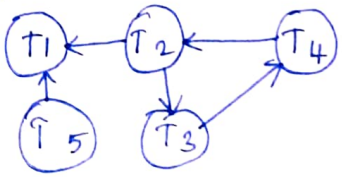


T_2 and T_3 appear in both sites indicating that they are requested items in both sites.

When transaction T_i on site S_1 need a resource in S_2, it sends a message of request to site2. If the resourse is held by transaction T_j, the system inerts and edge $T_i \rightarrow T_j$ in the local wait for graph of site S_2.

If any local-wait for graph has a cycle, then deadlock has occurred. Also no cycles in any of the wait for graphs doesn't mean that there are no deadlocks.

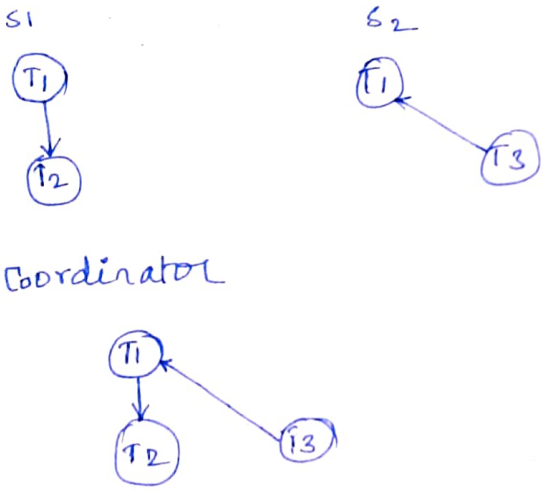
For example,



Each wait-for graph is acyclic, nevertheless, a deadlock exists in the system because the union of the ~~global~~ local wait-for graph contains a cycle as above.

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

False cycles in global wait for graph.



Suppose that T_2 releases the resource that is holding in site S_1, resulting in the deletion of edge $T_1 - T_2$ in S_2. The Transaction then requests a resource held by T_2 at site T_1, resulting in the addition of the edge $T_2 \rightarrow T_3$ in S_2.

Deadlock recovery may be initiated, although no deadlock has occured.