Name : Kamalesh. S Regn. No : 212720080104O Branch & Sec : I T A'

Subcode & Subname : IT18305 - Database Systems CAT-3

Part - b

6

(b) → When data in tables are stored across different sites, there is a high chance of data redundancy to occur. Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when same data exists in diff. formats in multiple tables. Data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information.

→ If data consistency is not followed, then there is a high chance of data corruption due to result of overwriting & overloading the same data again & again. If file gets corrupt, then the queries to retrieve the data won't work properly.

→ Data inconsistency also increase the database size & the complexity of the database. This is due to the repetition of the same data stored across multiple website again & again.

→ Data inconsistency also will lead up increasing the cost, as more cost would be required for a bigger database. To maintain such a large database is also a tough task.

To avoid these problems & maintain data consistency, we can follow the following methods :-

(i) Master data :-

It is a single source of common business data that is shared across several applications or systems. Although master data does not reduce the occurrences of data redundancy, it allows the company to work around & accept a certain level of data redundancy. This is because the use of master data ensures that in the event a data piece changes, an organisation only needs to update one piece of data. In this case, redundant data is consistently updated & provides the same info.

(ii) Database normalization :-

- Database normalization is the process of efficiently organising data in a database so that redundant data is eliminated. This process can ensure that all of a company's data looks & reads similarly across all records. By implementing

data normalization, an organization standardizes data fields such as customer name, addresses, & phone numbers.

- Normalizing data involves organising the columns & tables of a database to make sure their dependencies are enforced correctly. The "normal form" refers to the set of rules or normalizing data, & a database is known as 'normalized' if its free of delete, update & insert anomalities.

Requirements for managing data consistency :-

- high speed, low input data comparison.
- Support for heterogeneous databases.
- Capability for handling large data volumes.
- Data security.
- Easy to use, understand, configure, deploy & diagnose.
- Low impact on hardware & network resources.

Challenges in maintaing data consistency :-

→ migration errors :-
Different kinds of migration tools are employed to facilitate the initial load of the target database before replication can begin. Differences in config for handling data by the migration tools & replication products can result in data discrepancies.

→ Lift & shift workload to cloud :-
Since the world is moving towards cloud, the lift & shift of database workload from on-premises to cloud in need of today's IT world.

→ Differences in Source & target :-
Differences in source & target database config, for examp diff. encodings, locales, endianness or database versions can cause subtle discrepancies to happen during migration & replication.

→ Instantaneous Errors :-
Before migration or replication can begin, the target database will need to be instantiated with correct schema & constraints - Failure to do so will result in source & target being out of sync.

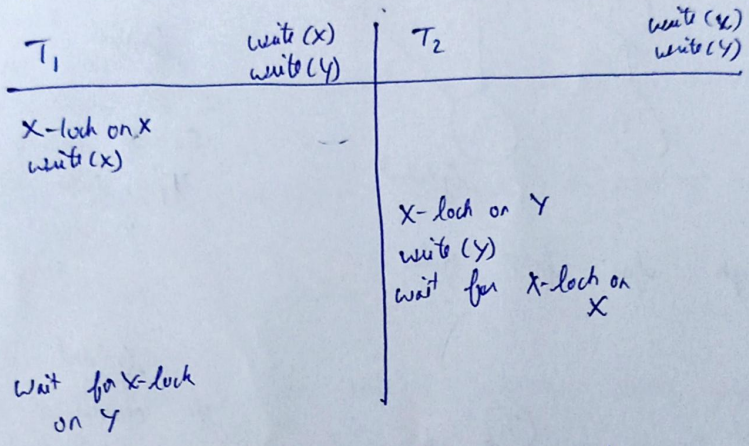
Part - C

8) b) → Distributed lock manager can be used for this scenario. In this approach, functionality of locking is implemented by lock managers at each site.

- Lock managers control access to local data items.
- Locking is performed separately on each site accessed by transaction.
  * Every replica must be locked & updated!
  * But special protocols may be used for replicas.

Deadlock handling

Consider the following 2 transactions & history, with item X & transaction T_1 at site 1, & item Y & transaction T_2 at site 2:

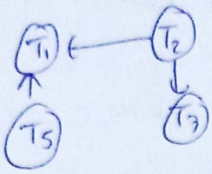


Result: Deadlock which can't be detected locally at either site.

Deadlock detection

- In the centralised deadlock-detection approach, global wait-for-graph is constructed & maintained in a single site:
  - Real graph: Real, but unknown state of system.
  - Constructed graph: Approximation generated by controller during execution of its algorithm.
- The global wait-for graph can be constructed when:
  - a new edge is inserted in or removed from one of local wait-for graphs.
  - a no. of changes have occured in a local wait-for graph.
  - the coordinator needs to invoke cycle-detector.
- If coordinator finds a cycle, it selects a victim & notifies all sites. The sites roll back the victim transaction.

Local & Global wait-for graphs

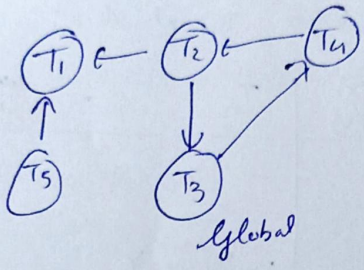


Site S_1

Site S_2

Local

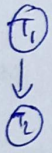
This fig. depicts a system of 2 sites, each maintaining its local wait-for graph. Transactions T_2 & T_3 appear in both graphs, indicating that transactions have requested items at both sites. When a transaction T_i on site S_1 needs a resource in site S_2, it sends a req msg to site S_2. If resource is held by transaction T_j, system inserts an edge $T_i - T_j$ in local wait-for graph of site S_2.



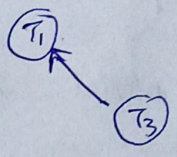
Global

Wait-for graph for false cycles

Initial state:



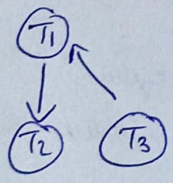
S_1



S_2

Global

In centralised deadlock detection approach, the system constructs & 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 b/w 2 types of wait-for graphs.



Coordinator

False cycles:-

- Suppose that starting from the state shown above.
  1) T_2 releases resource at S_1.
     → resulting in a message remove $T_1 \rightarrow T_2$ message from Transaction manager at site S_1 to coordinator
  2) T_2 requests a resource held by T_3 at site S_2
     → resulting in a message insert $T_2 \rightarrow T_3$ from S_2 to coordinator.
- Suppose further that insert message reaches before delete message.
  → This can happen due to network delays

- The coordinator would then find a false cycle

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_1$$

- The false cycle above never existed in reality.
- False cycles cannot occur if 2-phase locking is used.

Distributed deadlocks:-

- Unnecessary rollbacks may result
  - → when deadlock has indeed occured & a victim has been picked, & meanwhile one of transactions was aborted for reasons unrelated to the deadlock.
  - → Due to false cycles in global wait-for graph, however, likelihood of false cycle is low
- In distributed deadlock-detection approach, sites exchange wait-for info & check for deadlocks
  - → Expensive & not used in practice.

7)a) → Torrenting is, simply put, a file-sharing technology. Users, peers in this case, connect & share the files online. When you are using torrents you don't download files from a single website / source. Files circulate around the peers who seed them. It means that you download stuff from one direct source but you also download chunks from other users of same torrent. That enables a smooth transfer that's hard to trace.

→ Peers are users who are involved in the sharing process through torrent P2P network. You are a peer as long as you keep sharing file on the given network.

→ seeders are users who are downloading but also uploading the file for other users to download. When you download something through torrent, you download certain amounts of data but you also upload a bit. That upload means that other users will get chunk they need when you upload it.

→ Indexers are websites that work as search engines for files & content that can be downloaded through torrents.

Fault tolerance is used to ensure the working of torrent due to failure of server due to access from multiple geographical locations.

→ Torrent protocol enables decentralization of its resources by making use of

peer-to-peer network.

→ A small torrent file is created to represent a file or a folder to be shared.

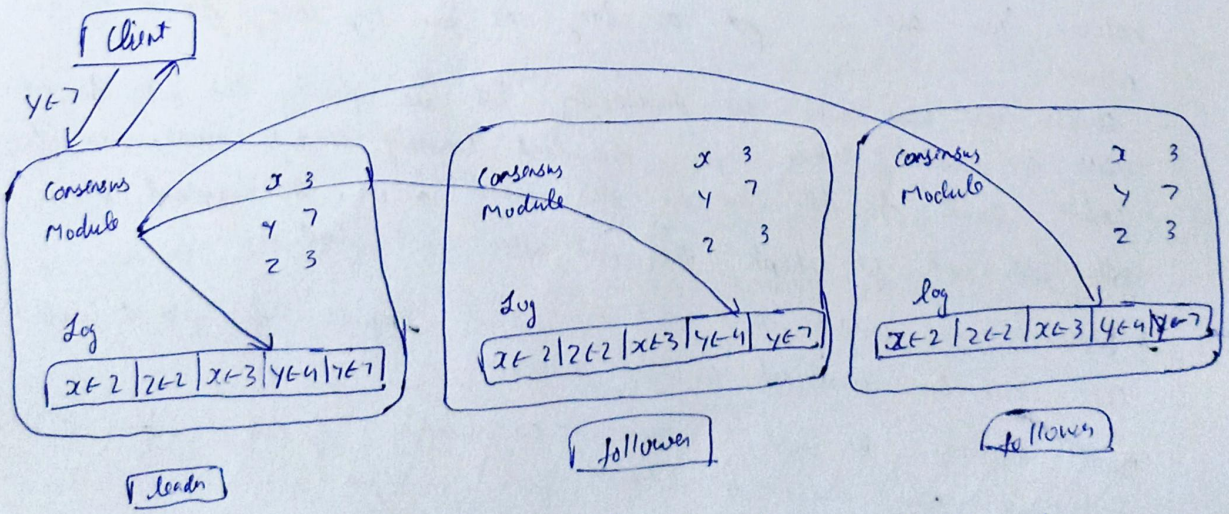
→ Users can download the required files using a unique magnet link associated to each file on torrent.

→ To learn the internet location of peers, we use distributed hash tables. The technology to ensure fault tolerance is Fault-tolerant services using replicated state machines.

Fault-Tolerant Services using replicated State machines

- Key requirement: makes the torrent service fault tolerant. Eg: lock manager, Key-value storage system.
- State machines are a powerful approach to contain such services.
- A State machine
  - → has a stored state, & receives inputs
  - → Makes state transitions on each input & may output some results
    - * transitions & output must be deterministic.
- A replicated state machine is a state machine that is replicated on multiple nodes
  - → All replicas must get exactly same inputs.
  - → Even if some of nodes fail, state & output can be obtained from other nodes

Replicated State Machine



Leader declares log declared 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.

Uses of Replicated State machines

- 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 key-value store:-

- State: key-value storage state
- Operations: get() & put()
- Operations executed when log is in committed state.

Part - A

3) Advantages of storing multiple relations in a single file:

- Complex structures can be implemented through the DBMS, thus increasing performance.
- Organise data & programs on storage media in a human friendly way.
- Editing data is easier.

Disadvantages:-

- Data backup: If the single file is corrupted, all data is lost, so no data backup.
- Increases size & complexity of DBMS.
- Data inconsistency

4)

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

In first case, A disk is failed for 100 hrs every 100 000 hrs. So in order to make B to fail, it will need $100000^{\wedge}2 / 100$ hrs. Because the other case, the time is reduced to $100000^{\wedge}2 (2 * 100)$

5) Map database management systems are software programs designed to efficiently store & recall spatial information. They are widely used in localisation & navigation, especially in automotive applications. A map database enables the rapid indexing & lookup of a large amount of geographic data.

4) MySQL enables restrictions to be placed on reuse of previous passwords. MySQL has password-reuse policy that allows it, to establish password-reuse policy globally, use password-history & password-reuse-interval system variables.

2) Database indexing. Hash tables may also be used as disk-based data structures & 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.