Database Management Systems

Distributed Databases

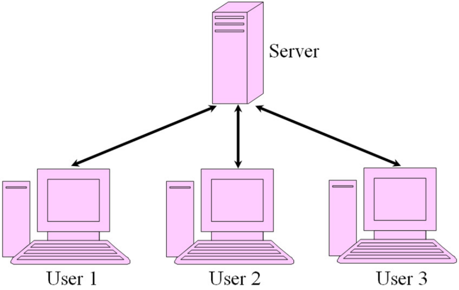
Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
and
Centre for Artificial Intelligence and Machine Learning
Indian Statistical Institute, Kolkata

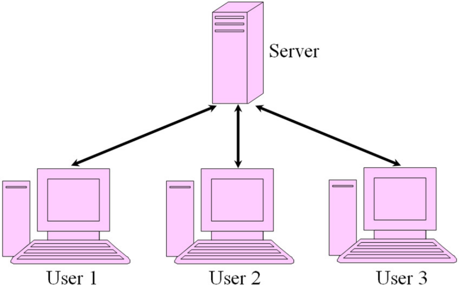
June, 2021

Basics



Centralized client-server architecture

Basics



Centralized client-server architecture

A distributed database system consists of loosely coupled sites that share no physical component. Database systems that run on each site are independent of each other. Transactions may access data at one or more sites.

Homogeneous and heterogeneous databases

In a homogeneous distributed database

– all sites have identical software

– all are aware of each other and agree to cooperate in processing user requests

– each site surrenders part of its autonomy in terms of right to change schemas or software

– the entire system appears as a single system to the user

Homogeneous and heterogeneous databases

In a homogeneous distributed database

– all sites have identical software

– all are aware of each other and agree to cooperate in processing user requests

– each site surrenders part of its autonomy in terms of right to change schemas or software

– the entire system appears as a single system to the user

In a heterogeneous distributed database

– different sites may use different schemas and software

– difference in schema is a major problem for query processing

– difference in software is a major problem for transaction processing

Data distribution

Data can be distributed in two ways:

- Replication – The system maintains several identical replicas (copies) of the relation, and stores each replica at a different site. The alternative to replication is to store only one copy of a relation.
- Fragmentation – The system partitions the relation into several fragments, and stores each fragment at a different site.

Data distribution

Data can be distributed in two ways:

- Replication – The system maintains several identical replicas (copies) of the relation, and stores each replica at a different site. The alternative to replication is to store only one copy of a relation.
- Fragmentation – The system partitions the relation into several fragments, and stores each fragment at a different site.

Note: The fragmentation can be lossless (original relation can be restored from the partitions) or lossy (original relation can not be restored from the partitions).

Data distribution

Replication	Fragmentation
Advantageous in terms of high availability	Might not be readily available
Advantageous in terms of time complexity but not space complexity	Maintains a balance between the time and space complexity
Disadvantageous in view of the redundancy and for updating	No redundancy or problem in updating

Data transparency

Data transparency denotes the degree to which a system user may remain unaware of the details of how and where the data items are stored in a distributed system.

Data transparency

Data transparency denotes the degree to which a system user may remain unaware of the details of how and where the data items are stored in a distributed system.

It can be of the following types:

1. Replication transparency – Users are not required to know what data objects have been replicated, or where replicas have been placed.
2. Fragmentation transparency – Users do not have to be concerned with how a relation has been fragmented.
3. Location transparency – Users are not required to know the physical location of the data.

Horizontal fragmentation

Name	Age	Area
Malay	38	Crowdsourcing
Ansuman	44	High Performance Architectures

Name	Age	Area
Sasthi	47	Wireless Networks
Sourav	40	Theoretical Computer Science

Horizontal fragmentation

Name	Age	Area
Malay	38	Crowdsourcing
Ansuman	44	High Performance Architectures

Name	Age	Area
Sasthi	47	Wireless Networks
Sourav	40	Theoretical Computer Science

Note: Horizontal fragmentation is lossless when union of the fragments produces the original relation.

Vertical fragmentation

Name	Age
Malay	38
Ansuman	44
Sasthi	47
Sourav	40

Area
Crowdsourcing
High Performance Architectures
Wireless Networks
Theoretical Computer Science

Vertical fragmentation

Name	Age
Malay	38
Ansuman	44
Sasthi	47
Sourav	40

Area
Crowdsourcing
High Performance Architectures
Wireless Networks
Theoretical Computer Science

Note: Vertical fragmentation is lossless when natural join of the fragments produces the original relation.

Advantages of horizontal and vertical fragmentation

Horizontal:

- It allows parallel processing on fragments of a relation.
- It allows a relation to be split so that tuples are located where they are most frequently accessed.

Advantages of horizontal and vertical fragmentation

Horizontal:

- It allows parallel processing on fragments of a relation.
- It allows a relation to be split so that tuples are located where they are most frequently accessed.

Vertical:

- It allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed.
- Here tuple-id attribute allows efficient joining of vertical fragments.
- It allows parallel processing on a relation.

Advantages of horizontal and vertical fragmentation

Horizontal:

- It allows parallel processing on fragments of a relation.
- It allows a relation to be split so that tuples are located where they are most frequently accessed.

Vertical:

- It allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed.
- Here tuple-id attribute allows efficient joining of vertical fragments.
- It allows parallel processing on a relation.

Vertical and horizontal fragmentation can be mixed (hybrid fragmentation) and the advantage is that fragments may be successively fragmented to an arbitrary depth.

Hybrid fragmentation

A hybrid fragment neither include all the tuples for an attribute (likewise vertical fragmentation) nor all the attributes for a tuple (likewise horizontal fragmentation).

Name	Age
Malay	38
Ansuman	44
Sourav	40

Distributed transaction management

- Transaction may access data at several sites.

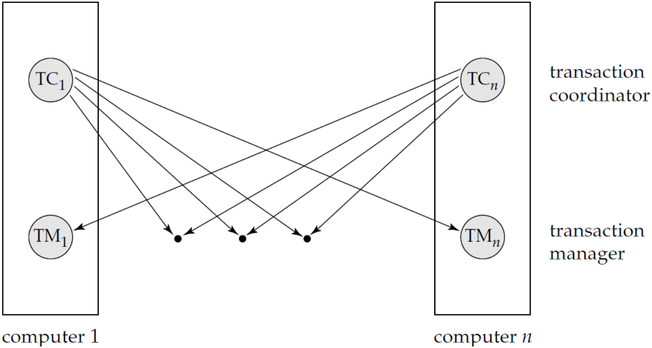
Distributed transaction management

- Transaction may access data at several sites.
- Each site has a local *transaction manager* responsible for:
  1. Maintaining a log for recovery purposes
  2. Participating in coordinating the concurrent execution of the transactions executing at that site.

Distributed transaction management

- Transaction may access data at several sites.
- Each site has a local *transaction manager* responsible for:
  1. Maintaining a log for recovery purposes
  2. Participating in coordinating the concurrent execution of the transactions executing at that site.
- Each site has a *transaction coordinator*, which is responsible for:
  1. Starting the execution of transactions that originate at the site.
  2. Distributing subtransactions at appropriate sites for execution.
  3. Coordinating the termination of each transaction that originates at the site, which may result in the transaction being committed at all sites or aborted at all sites.

Distributed transaction management



Distributed system architecture for transaction management

Locking protocols – Basics

The standard locking protocols used in a centralized system can also be used in a distributed environment. The only change that needs to be made is in the way the lock manager deals with replicated data.

We will consider the existence of `shared` and `exclusive` locking modes here.

Locking protocols – Single lock-manager

It works as follows.

1. The system maintains a single lock-manager residing in a single chosen site (say S_i). All the lock and unlock requests are made to S_i.
2. For locking a data item, a transaction sends a lock request to S_i. If the lock-manager grants the request immediately, it sends a message to the site at which the lock request was initiated. Otherwise, the request is delayed until it can be granted.
3. The transaction can read the data item from any one of the sites at which a replica of the data item resides. In the case of a write, all the sites where a replica of the data item resides must be involved in the writing.

Locking protocols – Single lock-manager

Advantages:

- The implementation is simple because it requires two messages for handling lock requests, and only one message for handling unlock requests.
- Since all the lock and unlock requests are made at one site, the standard deadlock-handling techniques can directly be applied.

Disadvantages:

- Since all the lock and unlock requests are processed on S_i, the site becomes a bottleneck.
- If the site S_i fails, the concurrency controller is lost.

Locking protocols – Distributed lock-manager

It works as follows.

1. Each site maintains a local lock-manager whose function is to administer the lock and unlock requests for those data items that are stored in that site.
2. When a transaction wishes to lock a data item Q, which is not replicated and resides at site S_i, a message is sent to the lock manager at site S_i requesting a lock (in a particular lock mode). If data item Q is locked in an incompatible mode, then the request is delayed until it can be granted. Once it has determined that the lock request can be granted, the lock manager sends a message back to the initiator indicating that it has granted the lock request.

Locking protocols – Primary copy

When a system uses data replication, we can choose one of the replicas as the primary copy. Thus, for each data item Q, the primary copy of Q must reside in precisely one site, which we call the primary site of Q.

When a transaction needs to lock a data item Q, it requests a lock at the primary site of Q. As before, the response to the request is delayed until it can be granted.

Locking protocols – Majority protocol

It works as follows.

1. If a data item Q is replicated in n different sites, then a lock-request message must be sent to more than one-half of the n sites in which Q is stored. Each lock manager determines whether the lock can be granted immediately (as far as it is concerned).
2. he response is delayed until the request can be granted. The transaction does not operate on Q until it has successfully obtained a lock on a majority of the replicas of Q.

Locking protocols – Biased protocol

It works as follows.

1. When a transaction needs to lock data item Q, it simply requests a lock on Q from the lock manager at one site that contains a replica of Q.
2. When a transaction needs to lock data item Q, it requests a lock on Q from the lock manager at all sites that contain a replica of Q.

Locking protocols – Quorum consensus protocol

The quorum consensus protocol is a generalization of the majority protocol. It works as follows.

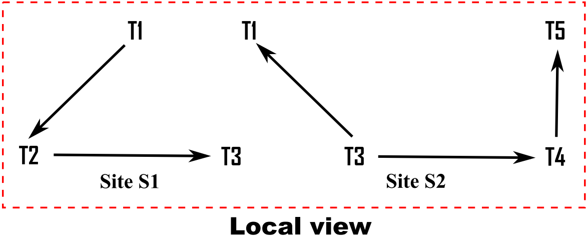
1. The quorum consensus protocol assigns each site a nonnegative weight. It assigns a pair of integers, called read quorum Q_r and write quorum Q_w, for read and write operations on a data item Q such that they satisfy the following conditions: (i) $Q_r + Q_w > S$ and (ii) $2Q_w > S$. Here, S is the total weight of all sites at which Q resides.
2. To execute a read operation, enough replicas must be read that their total weight is no less than Q_r.
3. To execute a write operation, enough replicas must be written so that their total weight is no less than Q_w.

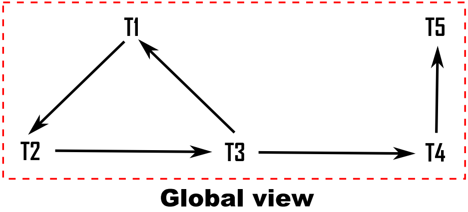
Locking protocols – Timestamping

There are two primary methods for generating unique timestamps, one centralized and one distributed.

- In the centralized scheme, a single site distributes the timestamps. The site can use a logical counter or its own local clock for this purpose.
- In the distributed scheme, each site generates a unique local timestamp by using either a logical counter or the local clock. We obtain the unique global timestamp by concatenating the unique local timestamp with the site identifier, which also must be unique. Note that, the order of concatenation is important.

Distributed deadlock handling





Local and global wait-for graphs