# Distributed Databases

CO527 Advanced Database Systems

1

## Overview

- Distributed Database Concepts
  - Advantages of Distributed Databases
- Data Fragmentation, Replication, and Allocation Techniques
- Concurrency Control and Recovery
- Distributed Transactions
  - Two-phase commit protocol
  - Three-phase commit protocol
- Distributed Query Processing
- Types of Distributed Database Systems

2

## Distributed Database Concepts

- A Distributed Database (DDB) is a collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
- Distributed Database is a combined result of the two technologies
  - Database technology
  - Network and Data Communication Technology

3

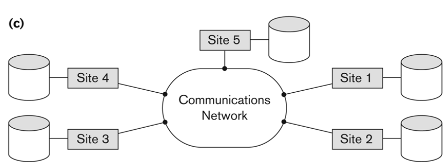
## Distributed Database Concepts (cont.)

- Recently distributed and database technologies are being developed for dealing with vast amount of data which is known as big data technologies.
- Then data mining and machine learning algorithms are used to extract needed knowledge.

4

## Advantages of Distributed Database System

- Management of distributed data with different **levels of transparency**:
  - This refers to the physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency/data organization transparency).

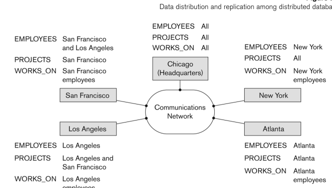



5

## Advantages of Distributed Database System

- Example
  - The EMPLOYEE, PROJECT, and WORKS_ON tables may be fragmented horizontally and stored with possible replication as shown below.

**Figure 25.2**
Data distribution and replication among distributed databases.




6

## Advantages of Distributed Database System

- **Distribution and Network transparency:**
  - Users do not have to worry about operational details of the network.
    - There is Location transparency, which refers to freedom of issuing command from any location without affecting its working.
    - Then there is Naming transparency, which allows access to any names object (files, relations, etc.) from any location.
- **Replication transparency**:
  - It allows to store copies of a data at multiple sites as shown in the above diagram.
  - This is done to minimize access time to the required data.
- **Fragmentation transparency**:
  - Allows to fragment a relation horizontally (create a subset of tuples of a relation) or vertically (create a subset of columns of a relation).

7

## Advantages of Distributed Database System

- Other Advantages
  - **Improved ease and flexibility of application development**
  - **Increased reliability and availability**:
    - Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
    - A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.
  - **Improved performance**:
    - A distributed DBMS fragments the database to keep data closer to where it is needed most.
    - This reduces data management (access and modification) time significantly.
  - **Easier expansion (scalability)**:
    - Horizontal scalability: Allows new nodes (computers) to be added anytime without chaining the entire configuration.
    - Vertical scalability: Expanding capacity of individual nodes of the system or processing power.

8

## Data Fragmentation, Replication and Allocation

- **Data Fragmentation**
  - Split a relation into logically related and correct parts. A relation can be fragmented in two ways:
    - **Horizontal Fragmentation**
    - **Vertical Fragmentation**

9

## Data Fragmentation, Replication and Allocation

- **Horizontal fragmentation (Sharding)**
  - It is a horizontal subset of a relation which contain those of tuples which satisfy selection conditions.
  - Consider the Employee relation with selection condition (DNO = 5). All tuples satisfy this condition will create a subset which will be a horizontal fragment of Employee relation.
  - A selection condition may be composed of several conditions connected by AND or OR.
  - Derived horizontal fragmentation: It is the partitioning of a primary relation to other secondary relations which are related with Foreign keys.

10

## Data Fragmentation, Replication and Allocation

- **Vertical fragmentation**
  - It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns. There is no selection condition used in vertical fragmentation.
  - Consider the Employee relation. A vertical fragment of can be created by keeping the values of Name, Bdate, Sex, and Address.
  - Because there is no condition for creating a vertical fragment, each fragment must include the primary key attribute of the parent relation Employee. In this way all vertical fragments of a relation are connected.

11

## Data Fragmentation, Replication and Allocation

- **Data Replication**
  - Database is replicated to all sites.
  - In full replication the entire database is replicated (called fully replicated distributed database) and in partial replication some selected part is replicated to some of the sites.
- **Data Distribution (Data Allocation)**
  - This is relevant only in the case of partial replication or partition.
  - The selected portion of the database is distributed to the database sites.

12

## Data Fragmentation, Replication and Allocation

(a) EMPD_5

| Fname | Minit | Lname | Ssn | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|
| John | B | Smith | 123456789 | 30000 | 333445555 | 5 |
| Franklin | T | Wong | 333445555 | 40000 | 888665555 | 5 |
| Ramesh | K | Narayan | 666884444 | 38000 | 333445555 | 5 |
| Joyce | A | English | 453453453 | 25000 | 333445555 | 5 |

DEP_5

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|
| Research | 5 | 333445555 | 1988-05-22 |

DEP_5_LOCS

| Dnumber | Location |
|---|---|
| 5 | Bellaire |
| 5 | Sugarland |
| 5 | Houston |

WORKS_ON_5

| Essn | Pno | Hours |
|---|---|---|
| 123456789 | 1 | 32.5 |
| 123456789 | 2 | 7.5 |
| 666884444 | 3 | 40.0 |
| 453453453 | 1 | 20.0 |
| 453453453 | 2 | 20.0 |
| 333445555 | 2 | 10.0 |
| 333445555 | 3 | 10.0 |
| 333445555 | 10 | 10.0 |
| 333445555 | 20 | 10.0 |

PROJS_5

| Pname | Pnumber | Plocation | Dnum |
|---|---|---|---|
| Product X | 1 | Bellaire | 5 |
| Product Y | 2 | Sugarland | 5 |
| Product Z | 3 | Houston | 5 |

Data at site 2

**Figure 25.8**
Allocation of fragments to sites. (a) Relation fragments at site 2 corresponding to department 5. (b) Relation fragments at site 3 corresponding to department 4.

13

## Data Fragmentation, Replication and Allocation

(b) EMPD_4

| Fname | Minit | Lname | Ssn | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|
| Alicia | J | Zelaya | 999887777 | 25000 | 987654321 | 4 |
| Jennifer | S | Wallace | 987654321 | 43000 | 888665555 | 4 |
| Ahmad | V | Jabbar | 987987987 | 25000 | 987654321 | 4 |

DEP_4

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|
| Administration | 4 | 987654321 | 1995-01-01 |

DEP_4_LOCS

| Dnumber | Location |
|---|---|
| 4 | Stafford |

WORKS_ON_4

| Essn | Pno | Hours |
|---|---|---|
| 333445555 | 10 | 10.0 |
| 999887777 | 30 | 30.0 |
| 999887777 | 10 | 10.0 |
| 987987987 | 10 | 35.0 |
| 987987987 | 30 | 5.0 |
| 987654321 | 30 | 20.0 |
| 987654321 | 20 | 15.0 |

PROJS_4

| Pname | Pnumber | Plocation | Dnum |
|---|---|---|---|
| Computerization | 10 | Stafford | 4 |
| New_benefits | 30 | Stafford | 4 |

Data at site 3

14

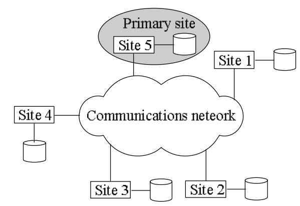
## Concurrency Control and Recovery

- Dealing with multiple copies of data items:
  - The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.
- Failure of individual sites:
  - Database availability must not be affected due to the failure of one or two sites and the recovery scheme must recover them before they are available for use.
- Communication link failure:
  - This failure may create network partition which would affect database availability even though all database sites may be running.
- Distributed commit:
  - A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit.
- Distributed deadlock:
  - Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.

16

## Concurrency Control and Recovery

- Distributed Concurrency control based on a distributed copy of a data item
  - **Primary site technique**: A single site is designated as a primary site which serves as a coordinator for transaction management.




17

## Concurrency Control and Recovery

- Transaction management:
  - Concurrency control and commit are managed by this site.
  - In two phase locking, this site manages locking and releasing data items. If all transactions follow two-phase policy at all sites, then serializability is guaranteed.

18

## Concurrency Control and Recovery

- Transaction Management
  - Advantages:
    - An extension to the centralized two phase locking so implementation and management is simple.
    - Data items are locked only at one site but they can be accessed at any site.
  - Disadvantages:
    - All transaction management activities go to primary site which is likely to overload the site.
    - If the primary site fails, the entire system is inaccessible.
  - To aid recovery a backup site is designated which behaves as a shadow of primary site. In case of primary site failure, backup site can act as primary site.

19

## Concurrency Control and Recovery

- Primary Copy Technique:
  - In this approach, instead of a site, a data item partition is designated as primary copy. To lock a data item just the primary copy of the data item is locked.
- Advantages:
  - Since primary copies are distributed at various sites, a single site is not overloaded with locking and unlocking requests.
- Disadvantages:
  - Identification of a primary copy is complex. A distributed directory must be maintained, possibly at all sites.

20

## Concurrency Control and Recovery

- Recovery from a coordinator failure
  - In both approaches a coordinator site or copy may become unavailable. This will require the selection of a new coordinator.
- Primary site approach with no backup site:
  - Aborts and restarts all active transactions at all sites. Elects a new coordinator and initiates transaction processing.
- Primary site approach with backup site:
  - Suspends all active transactions, designates the backup site as the primary site and identifies a new back up site. Primary site receives all transaction management information to resume processing.
- Primary and backup sites fail or no backup site:
  - Use election process to select a new coordinator site.

21

## Concurrency Control and Recovery

- Concurrency control based on voting:
  - There is no primary copy of coordinator.
  - Send lock request to sites that have data item.
  - If majority of sites grant lock then the requesting transaction gets the data item.
  - Locking information (grant or denied) is sent to all these sites.
  - To avoid unacceptably long wait, a time-out period is defined. If the requesting transaction does not get any vote information then the transaction is aborted.

22

## Distributed Transactions

- Transaction that updates data on two or more systems
- Challenge
  - Handle machine, software, & network failures while preserving transaction integrity

23

## Distributed Transactions

- Each computer runs a transaction manager
  - Responsible for subtransactions on that system
  - Transaction managers communicate with other transaction managers
  - Performs prepare, commit, and abort calls for subtransactions
- Every subtransaction must agree to commit changes before the transaction can complete

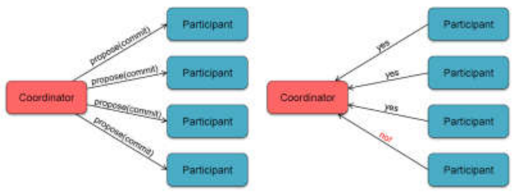
24

## Transactional Commits are Consensus

- Remember consensus?
  - Agree on a value proposed by at least one process
- The coordinator proposes to commit a transaction
  - Participants will agree ⇒ all participants then commit
  - Participants will not agree ⇒ all participants then abort
- Here, we need unanimous agreement to commit

25

## Two-Phase Commit Protocol



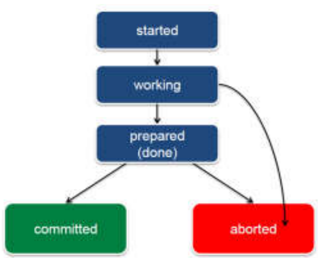

26

## Two-phase commit protocol

- Goal:
  - *Reliably agree to commit or abort a collection of sub-transactions*
- All processes in the transaction will agree to commit or abort
- One transaction manager is elected as a coordinator – the rest are participants
- Assume:
  - write-ahead log in stable storage
  - No system dies forever
  - Systems can always communicate with each other

27

## Transaction States

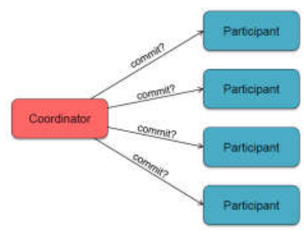



When a participant enters the *prepared state, it contacts the coordinator to* start the commit protocol to commit the entire transaction

28

## Two-Phase Commit Protocol

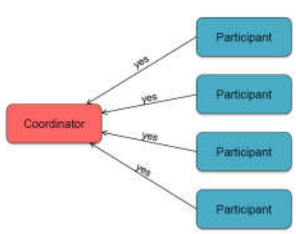
- Phase 1: Voting Phase
  - Get commit agreement from *every participant*




29

## Two-Phase Commit Protocol

- Phase 1: Voting Phase
  - Get commit agreement from *every participant*

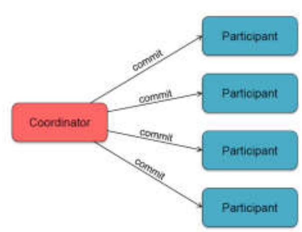



A single "no" response means that we will have to abort the transaction

30

## Two-Phase Commit Protocol

- Phase 2: Commit Phase
  - Send the results of the vote to every participant

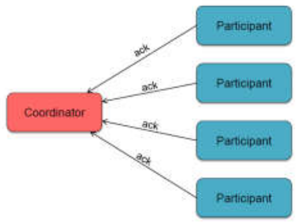



Send abort if any participant voted "no"

31

## Two-Phase Commit Protocol

- Phase 2: Commit Phase
  - Get “I have committed” acknowledgements from *every participant*




32

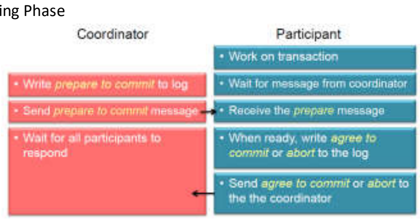
## Dealing with failure

- 2PC assumes a fail-recover model
  - Any failed system will eventually recover
- A recovered system cannot change its mind
  - If a node agreed to commit and then crashed, it must be willing and able to commit upon recovery
- Each system will use a writeahead (transaction) log
  - Keep track of where it is in the protocol (and what it agreed to)
  - As well as values to enable commit or abort (rollback)

33

## Two-Phase Commit Protocol: Phase 1

- Voting Phase

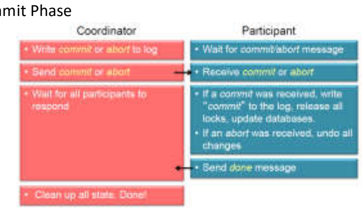



- Get distributed agreement: the coordinator asked each participant if it will commit or abort and received replies from each participant.

34

## Two-Phase Commit Protocol: Phase 2

- Commit Phase




- Tell *all participants to commit or abort*
- Get everyone’s response that they’re done

35

## Dealing with failure

- Failure during Phase 1 (voting)
  - Coordinator dies
    - Some participants may have responded; others have no clue
      - Coordinator restarts; checks log; sees that voting was in progress
      - Coordinator restarts voting
  - Participant dies
    - The participant may have died before or after sending its vote to the coordinator
      - If the coordinator received the vote, wait for other votes and go to phase 2
      - Otherwise: wait for the participant to recover and respond (keep querying it)

36

## Dealing with failure

- Failure during Phase 2 (commit/abort)
  - Coordinator dies
    - Some participants may have been given commit/abort instructions
      - Coordinator restarts; checks log; informs all participants of chosen action
  - Participant dies
    - The participant may have died before or after getting the commit/abort request
      - Coordinator keeps trying to contact the participant with the request
      - Participant recovers; checks log; gets request from coordinator
        - If it committed/aborted, acknowledge the request
        - Otherwise, process the commit/abort request and send back the acknowledgement

37

## Adding a recovery coordinator

- Another system can take over for the coordinator
  - Could be a participant that detected a timeout to the coordinator
- Recovery node needs to find the state of the protocol
  - Contact ALL participants to see how they voted
  - If we get voting results from all participants
    - We know that Phase 1 has completed
    - If all participants voted to commit ⇒ send commit request
    - Otherwise send abort request
  - If ANY participant has not voted
    - We know that Phase 1 has *not completed*
      - Restart the protocol
- But … if a participant node also crashes, we're stuck!
  - Have to wait for recovery

38

## What's wrong with the 2PC protocol?

- Biggest problem: it's a blocking protocol
  - If the coordinator crashes, participants have no idea whether to commit or abort
    - A recovery coordinator helps in some cases
  - A non-responding participant will also result in blocking
- When a participant gets a commit/abort message, it does not know if every other participant was informed of the result

39

## Three-Phase Commit Protocol

- Same setup as the two-phase commit protocol:
  - Coordinator & Participants
- Add timeouts to each phase that result in an abort
- Propagate the result of the commit/abort vote to each participant *before telling them to act on it*
  - This will allow us to recover the state if any participant dies

40

## Three-Phase Commit Protocol

- Split the second phase of 2PC into two parts:
  - 2a. *"Precommit" (prepare to commit) phase*
    - Send *Prepare message to all participants when it received a yes from all* participants in phase 1
    - Participants can prepare to commit but cannot do anything that cannot be undone
    - Participants reply with an acknowledgement
    - Purpose: *let every participant know the state of the result of the vote so that state can be recovered if anyone dies*
  - 2b. *"Commit" phase (same as in 2PC)*
    - If coordinator gets ACKs for all *"precommit" messages*
      - It will send a *commit message*
    - Else it will abort – send an *abort message to all participants*

41

## Three-Phase Commit Protocol

- Phase 1: *voting phase*
  - Coordinator sends *canCommit? queries to participants & et responses*
  - Purpose: Find out if everyone agrees to commit
  - It the coordinator gets a *timeout from any participant, or any NO replies are received*
    - Send an *abort to all participants*
  - *If a participant times out waiting for a request from the coordinator*
    - *It aborts itself (assume coordinator crashed)*
  - *Else continue to phase 2*
- *Phase 2: Prepare to commit phase*
  - *Send a Prepare message to* all participants. Get OK messages from all participants
  - Purpose: let all participants know the decision to commit
  - If coordinator times out: assume participant crashed, *send abort to all participants*
    - *The coordinator cannot count on every participant having received the Prepare message*
- Phase 3: finalize
  - Send commit messages to participants and get responses from all
  - If participant times *out: contact any other participant* and move to that state (commit or abort)
  - If coordinator times *out: that's ok*

42

## 3PC Recovery

- If the coordinator crashes
  - A recovery node can query the state from any available participants
- Possible states that the participant may report:
  - Already committed
    - That means that every other participant has received a *Prepare to Commit*
    - Some participants may have committed
    - Send *Commit message to all participants (just in case they didn't get it)*
  - Not committed but received a *Prepare message*
    - That means that all participants agreed to commit; some may have committed
    - Send *Prepare to Commit message to all participants (just in case they didn't get it)*
    - Wait for everyone to acknowledge; then commit
  - Not yet received a *Prepare message*
    - This means no participant has committed; some may have agreed
    - Transaction can be aborted or the commit protocol can be restarted

43

## 3PC Weaknesses

- Main weakness of 3PC
  - May have problems when the network gets partitioned
  - Partition A: nodes that received *Prepare message*
    - Recovery coordinator for A: allows commit
  - Partition B: nodes that did not receive *Prepare message*
    - Recovery coordinator for B: aborts
  - Either of these actions are legitimate as a whole
    - But when the network merges back, the system is inconsistent
- Not good when a crashed coordinator recovers
  - It needs to find out that someone took over and stay quiet
  - Otherwise it will mess up the protocol, leading to an inconsistent state

44

## 3PC coordinator recovery problem

- Suppose a coordinator sent a Prepare messages to all participants
- Suppose all participants acknowledged these message
  - BUT the coordinator died before it got all acknowledgements
- A recovery coordinator queries a participant
  - Continues with the commit: Sends Prepare, gets ACKs, sends Commit
- At the same time...
  - The original coordinator recovers
  - Realizes it is still missing some replies from the Prepare
  - Times out and decides to send an Abort to all participants
- Some processes may commit while others abort!
- 3PC is not resilient against asynchronous fail-recover failures

45

## Query Processing in Distributed Databases

- Issues
  - Cost of transferring data (files and results) over the network.
    - This cost is usually high so some optimization is necessary.
    - Example relations: Employee at site 1 and Department at Site 2
      - Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = $10^6$ bytes.

| Fname | Minit | Lname | SSN | Bdate | Address | Sex | Salary | Superssn | Dno |
|---|---|---|---|---|---|---|---|---|---|

      - Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.

| Dname | Dnumber | Mgrssn | Mgrstartdate |
|---|---|---|---|

    - Q: For each employee, retrieve employee name and department name Where the employee works.
    - Q: $\Pi_{Fname,Lname,Dname}$ (Employee $\bowtie_{Dno = Dnumber}$ Department)

46

## Query Processing in Distributed Databases

- Result
  - The result of this query will have 10,000 tuples, assuming that every employee is related to a department.
  - Suppose each result tuple is 40 bytes long. The query is submitted at site 3 and the result is sent to this site.
  - Problem: Employee and Department relations are not present at site 3.


47

## Query Processing in Distributed Databases

- Strategies:
  1. Transfer Employee and Department to site 3.
     - Total transfer bytes = 1,000,000 + 3500 = 1,003,500 bytes.
  2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3.
     - Query result size = 40 * 10,000 = 400,000 bytes. Total transfer size = 400,000 + 1,000,000 = 1,400,000 bytes.
  3. Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.
     - Total bytes transferred = 400,000 + 3500 = 403,500 bytes.
- Optimization criteria: minimizing data transfer.
  - Preferred approach: strategy 3.

48

## Query Processing in Distributed Databases

- Consider the query
  - Q': For each department, retrieve the department name and the name of the department manager
- Relational Algebra expression:
  - $\Pi_{Fname,Lname,Dname}$ (Employee $\bowtie_{Mgrssn = SSN}$ Department)

49

## Query Processing in Distributed Databases

- The result of this query will have 100 tuples, assuming that every department has a manager, the execution strategies are:
  1. Transfer Employee and Department to the result site and perform the join at site 3.
     - Total bytes transferred = 1,000,000 + 3500 = 1,003,500 bytes.
  2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3. Query result size = 40 * 100 = 4000 bytes.
     - Total transfer size = 4000 + 1,000,000 = 1,004,000 bytes.
  3. Transfer Department relation to site 1, execute join at site 1 and send the result to site 3.
     - Total transfer size = 4000 + 3500 = 7500 bytes.
- Preferred strategy: Choose strategy 3.

50

## Query Processing in Distributed Databases

- Now suppose the result site is 2. Possible strategies :
  1. Transfer Employee relation to site 2, execute the query and present the result to the user at site 2.
     - Total transfer size = 1,000,000 bytes for both queries Q and Q'.
  2. Transfer Department relation to site 1, execute join at site 1 and send the result back to site 2.
     - Total transfer size for Q = 400,000 + 3500 = 403,500 bytes and for Q' = 4000 + 3500 = 7500 bytes.

51

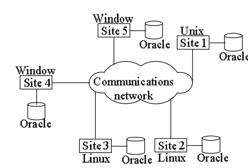
## Query Processing in Distributed Databases

- Semijoin:
  - Objective is to reduce the number of tuples in a relation before transferring it to another site.
- Example execution of Q or Q':
  1. Project the join attributes of Department at site 2, and transfer them to site 1. For Q, 4 * 100 = 400 bytes are transferred and for Q', 9 * 100 = 900 bytes are transferred.
  2. Join the transferred file with the Employee relation at site 1, and transfer the required attributes from the resulting file to site 2. For Q, 34 * 10,000 = 340,000 bytes are transferred and for Q', 39 * 100 = 3900 bytes are transferred.
  3. Execute the query by joining the transferred file with Department and present the result to the user at site 2.

52

## Types of Distributed Database Systems

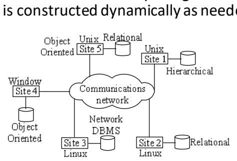
- Homogeneous
  - All sites of the database system have identical setup, i.e., same database system software.
  - The underlying operating system may be different.
    - For example, all sites run Oracle or DB2, or Sybase or some other database system.
  - The underlying operating systems can be a mixture of Linux, Window, Unix, etc.




53

## Types of Distributed Database Systems

- Heterogeneous
  - Federated: Each site may run different database system but the data access is managed through a single conceptual schema.
    - This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.
  - Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.




54

## Types of Distributed Database Systems

- Federated Database Management Systems Issues
  - Differences in data models:
    - Relational, Objected oriented, hierarchical, network, etc.
  - Differences in constraints:
    - Each site may have their own data accessing and processing constraints.
  - Differences in query language:
    - Some site may use SQL, some may use SQL-89, some may use SQL-92, and so on.

55