UWE Bristol | University of the West of England

Presented by

Dr. Trupti Padiya

School of Computing and Creative Technologies

Advanced Databases UFCFU3-15-3

Distributed Databases

Slides with modification adapted from Dr. Elias Pimenidis

Contents

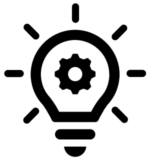
- Centralised databases
  - Advantages
  - Disadvantages

- Distributed databases
  - Why do we need them
  - Basic definition(s)
  - Conceptual understanding
  - Advantages
  - Disadvantages

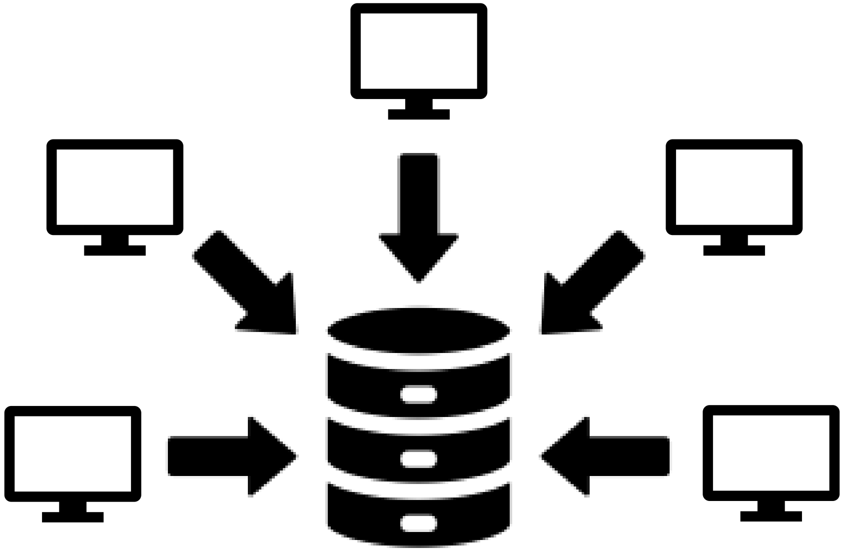
Centralised Databases

- Data is located, stored, and maintained in a single location

- Advantages
  - Easy to manage, update, backup…
  - Get a complete view of the data



Think of any other advantages of a centralised database/system

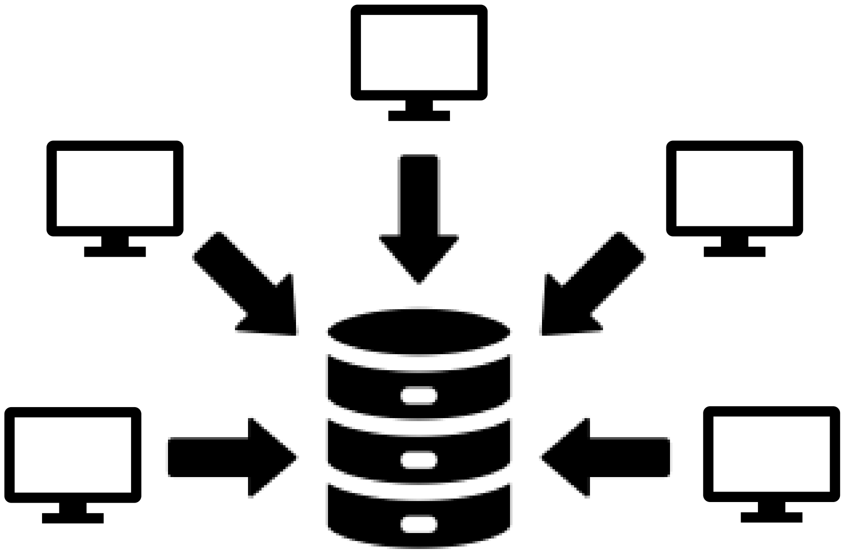


Centralised Database

Why do we need Distributed databases?: **Revisiting Centralized databases**

- Higher data traffic
- Access Speed
- System Failure
- Data Loss
- Difficult to Scale

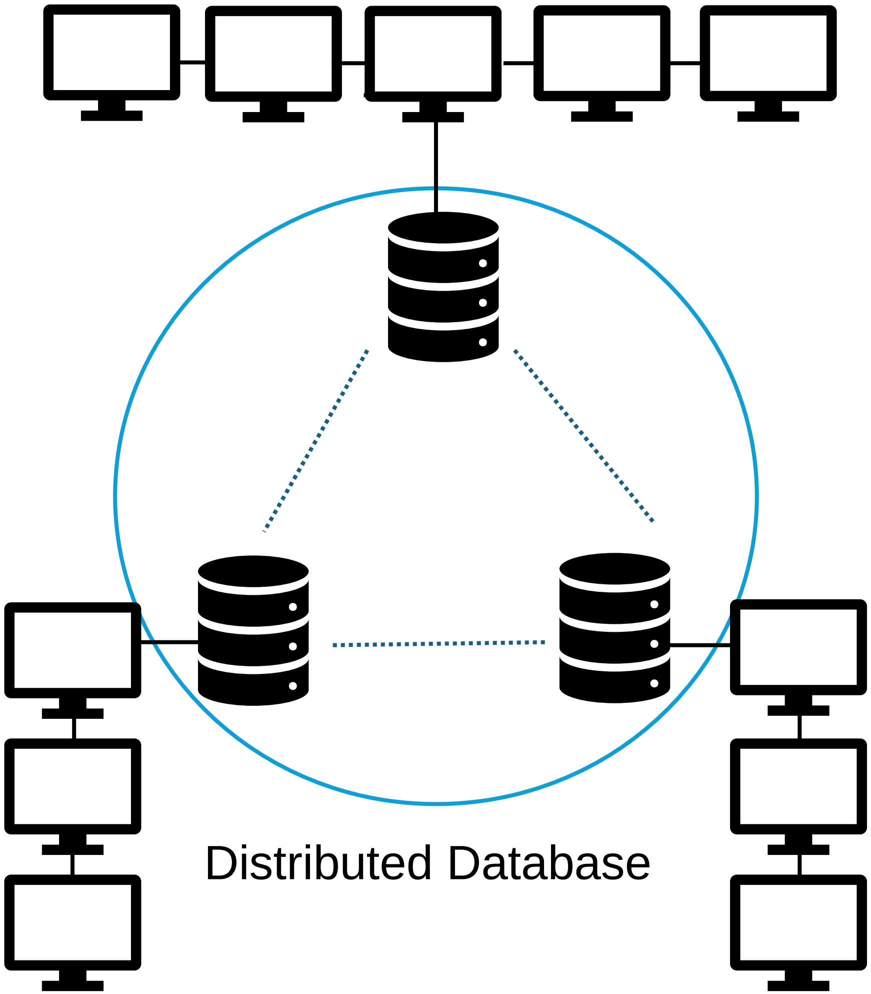
Think of any other disadvantages of a centralised database/system



Centralised Database

Distributed Databases

- ❑ A distributed database is a database in which data is stored across different physical locations.
- ❑ In the most basic terms, a distributed database is a database that stores data in multiple locations instead of one location (MongoDB)



Distributed database management system: **Benefits**

- **Flexible**
- **Resilience**
- **Scalable**
- **Improved performance.**
- **High availability**

Distributed database management system : **Challenges**

- **Complexity**
- **Latency**
- **Data consistency**
- **Cost**

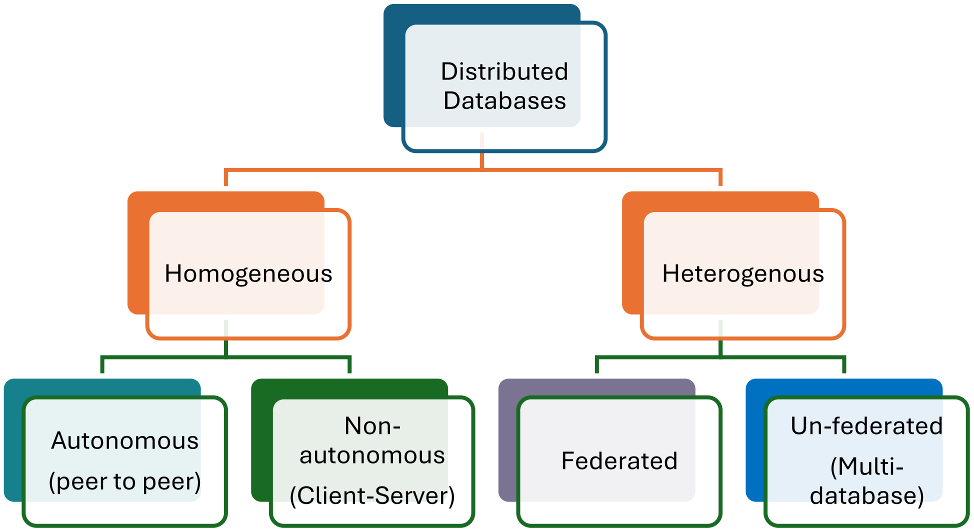
Advantages and Disadvantages of Distributed Databases

ADVANTAGES	DISADVANTAGES
Reflects organizational structure	Complexity
Improved shareability and local autonomy	Cost
Improved availability	Security
Improved reliability	Integrity control more difficult
Improved performance	Lack of standards
Economics	Lack of experience
Modular growth	Database design more complex
Integration	
Remaining competitive	

Reflect on advantages and disadvantages of a Distributed database/system. Try to relate it with some examples.

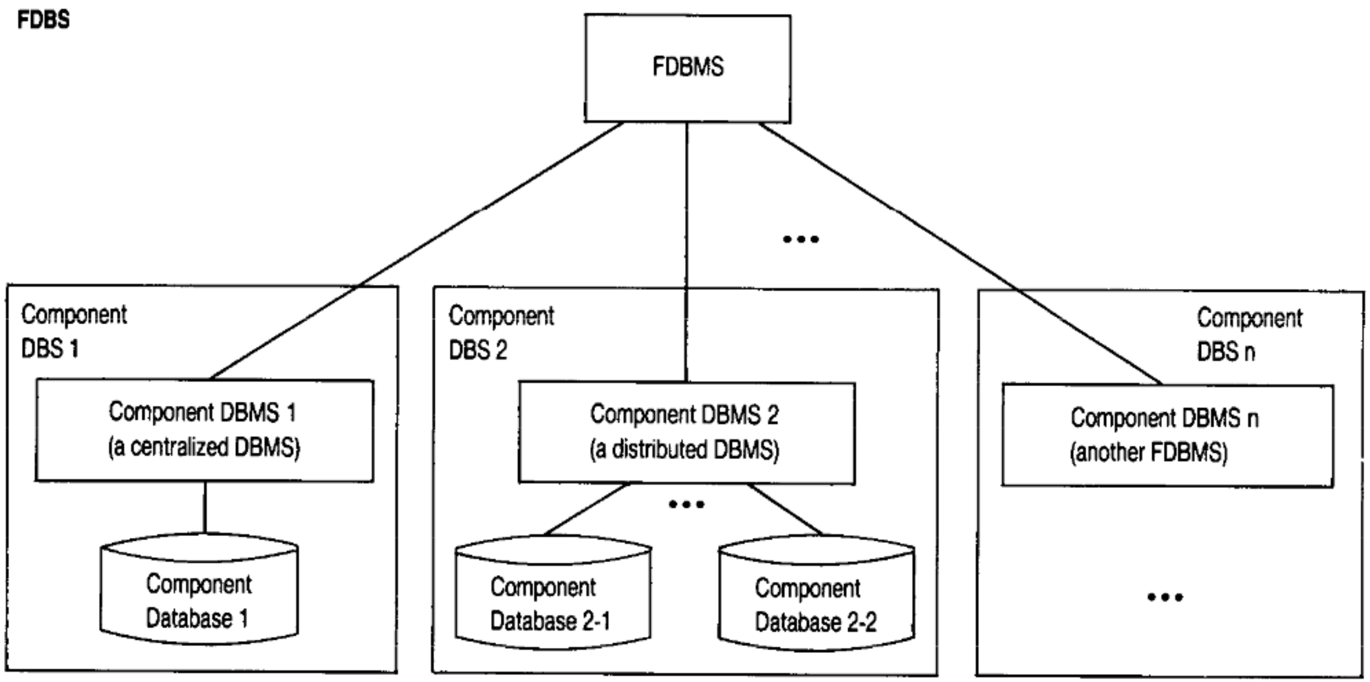
Types of Distributed Databases

- **Homogeneous:** *Every site runs same type of Database*
  - **Autonomous:** Every database is independent and functions on its own. They use message passing to share data updates.
  - **Non-Autonomous:** Data is distributed across the nodes and a central DBMS co-ordinates data updates across the sites.
- **Heterogenous:** *Different sites run different databases (even non-relational databases)*
  - **Federated (single schema):** *Each site may run a different database system, but the data access is managed through a single conceptual schema*
  - **Multi-database (No global schema):** *There is no conceptual global schema. For data access, a schema is constructed dynamically as needed by the application software*



Think of Homogeneous architecture designs that you might have learned/ implemented e.g. Peer to peer and client-server-based architectures. Also, think of some examples.

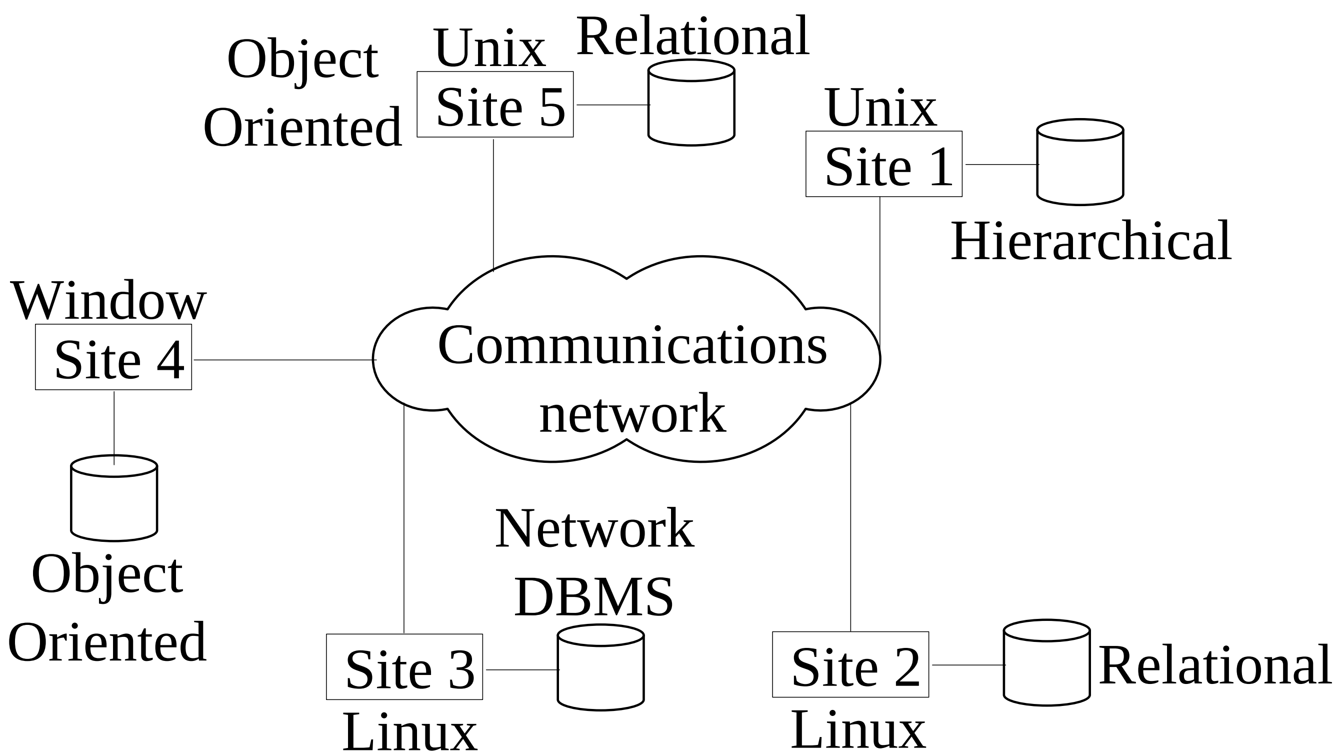
Heterogeneous DDBMS: **Federated Database**



FDBMS and its components (Sheth, A.P and Larson J, 1990))

Sheth, A.P. and Larson, J.A., 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys (CSUR), 22(3), pp.183-236.

Heterogeneous DDBMS: **Un-Federated Database (Multi database)**



Multi database and its components (Pimenidis E, 2023)

Pimenidis, E. 2023. Distributed Database Lecture 2023-24, Available at: https://blackboard.uwe.ac.uk/ultra/courses/_358500_1/cl/outline Accessed (Aug 2024)

Design for DDMBS

- Non-fragmented and Non-replicated
- Fragmentation
  - Vertical Fragmentation
  - Horizontal Fragmentation
  - Hybrid Fragmentation
- Replication
  - Full replication
  - Partial replication
- Mixed

Fragmentation

- A relation may be divided into a number of sub-relations, called fragments, which are then distributed.

- **Types of Fragmentation**
  - Vertical
  - Horizontal
  - Hybrid/Mixed

Vertical Fragmentation

Student Id	First Name	Last Name	Date of Birth	Program	Year	Pathway
202321001	Alice1	Pitman11	25/4/1998	P1	Y1	P1- Path1
202321003	Bob1	Franklin7	12/9/2002	P1	Y2	P1- Path2
202331011	Peter2	Fernandez4	3/7/2001	P10	Y1	P10- Path3
202321017	Sam3	Sharapov2	5/4/2002	P1	Y3	P1- Path1
202335001	Tom6	Graff1	6/2/2003	P5	Y1	P5- Path2

Student Id	First Name	Last Name	Date of Birth
202321001	Alice1	Pitman11	25/4/1998
202321003	Bob1	Franklin7	12/9/2002
202331011	Peter2	Fernandez4	3/7/2001
202321017	Sam3	Sharapov2	5/4/2002
202335001	Tom6	Graff1	6/2/2003

Student Id	Program	Year	Pathway
202321001	P1	Y1	P1- Path1
202321003	P1	Y2	P1- Path2
202331011	P10	Y1	P10- Path3
202321017	P1	Y3	P1- Path1
202335001	P5	Y1	P5- Path2

Think of other ways to vertically partition this table

Horizontal Fragmentation

Student Id	First Name	Last Name	Date of Birth	Program	Year	Pathway
202321001	Alice1	Pitman11	25/4/1998	P1	Y1	P1- Path1
202321003	Bob1	Franklin7	12/9/2002	P1	Y2	P1- Path2
202331011	Peter2	Fernandez4	3/7/2001	P10	Y1	P10- Path3
202321017	Sam3	Sharapov2	5/4/2002	P1	Y3	P1- Path1
202335001	Tom6	Graff1	6/2/2003	P5	Y1	P5- Path2

Student Id	First Name	Last Name	Date of Birth	Program	Year	Pathway
202321001	Alice1	Pitman11	25/4/1998	P1	Y1	P1- Path1
202321003	Bob1	Franklin7	12/9/2002	P1	Y2	P1- Path2
202321017	Sam3	Sharapov2	5/4/2002	P1	Y3	P1- Path1

Student Id	First Name	Last Name	Date of Birth	Program	Year	Pathway
202331011	Peter2	Fernandez4	3/7/2001	P10	Y1	P10- Path3

Student Id	First Name	Last Name	Date of Birth	Program	Year	Pathway
202335001	Tom6	Graff1	6/2/2003	P5	Y1	P5- Path2

Think of other ways to horizontally partition this table

Hybrid/Mixed Fragmentation

Student Id	First Name	Last Name	Date of Birth	Program	Year	Pathway
202321001	Alice1	Pitman11	25/4/1998	P1	Y1	P1- Path1
202321003	Bob1	Franklin7	12/9/2002	P1	Y2	P1- Path2
202331011	Peter2	Fernandez4	3/7/2001	P10	Y1	P10- Path3
202321017	Sam3	Sharapov2	5/4/2002	P1	Y3	P1- Path1
202335001	Tom6	Graff1	6/2/2003	P5	Y1	P5- Path2

Student Id	Program	Year	Pathway
202321001	P1	Y1	P1- Path1
202321003	P1	Y2	P1- Path2
202321017	P1	Y3	P1- Path1

Student Id	First Name	Last Name	Date of Birth
202321001	Alice1	Pitman11	25/4/1998
202321003	Bob1	Franklin7	12/9/2002
202331011	Peter2	Fernandez4	3/7/2001
202321017	Sam3	Sharapov2	5/4/2002
202335001	Tom6	Graff1	6/2/2003

Student Id	Program	Year	Pathway
202331011	P10	Y1	P10- Path3

Student Id	Program	Year	Pathway
202335001	P5	Y1	P5- Path2

Think of other ways to partition this table using hybrid approach

Fragmentation

- **Fragmentation Benefits**
  - Data locality
  - Smaller fragments - better query performance
  - Indexes smaller
  - Load balanced across nodes

Critically think of challenges associated with fragmentation. There are a couple of challenges – Hint – query joins (relational databases), speed, backups, …

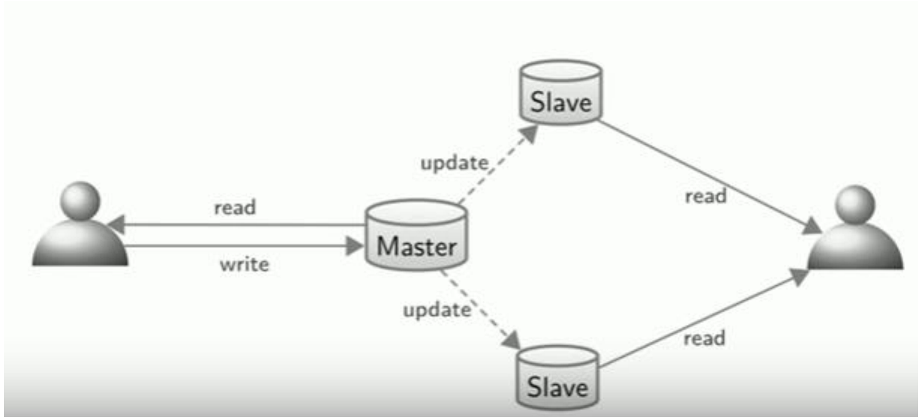
Replication

- **Replication – storing separate copies of databases across different (generally two or more) sites**

- **Types of replications**
  - **Full**
  - **Partial**

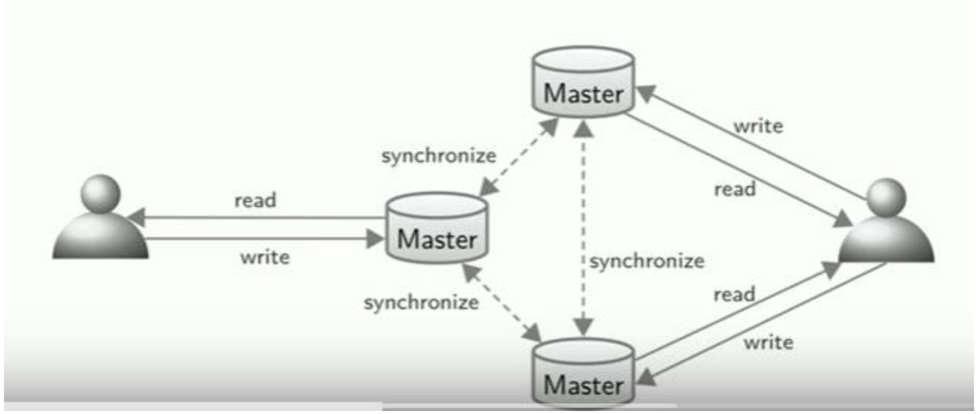
Master-Slave Replication

- Updates performed at a single master (Primary)
- At slave sites, data may only be read (Secondary)



Multi-Master Replication

- Updates permitted at any server holding a replica
- Automatic propagation to all replicas



Replication Strategies

- Asynchronous Replication (lazy replication) :
  - Replication performed **periodically**
  - different copies may get out of sync in the meantime.
- Synchronous (Eager Replication)
  - All replicas updated before the commit in a **single transaction**

Replication benefits and Challenges

- Higher data availability
- Reduced server load
- More reliable data
- Less data movement
- Better protection
- Lower latency
- Better application performance

Critically think of challenges associated with replication. There are a couple of challenges - Hint –consistency, concurrency, …

DDBMS: Transparency

- The DBMS is expected to be distribution transparent (invisible) to the user.
  - Objective of transparency: to make the distributed system appear like a centralized system.

- Types of Transparency
  - Location transparency
  - Fragmentation transparency
  - Replication transparency
  - Access transparency
  - Transaction transparency

Location transparency

- Users do not know where required data resides
  - Location transparency, refers to freedom of issuing command from any location without affecting its working.
  - You do not need to care about where the database tables (fragments) are located (You can execute any query from anywhere using the same schema)

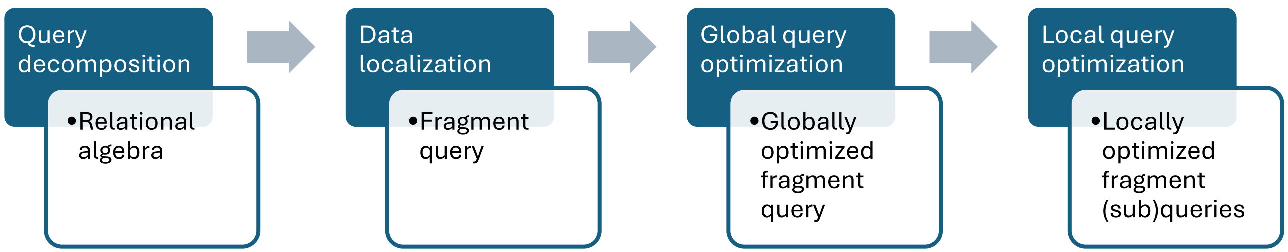
Fragmentation Transparency

- Users can execute global queries, without being concerned with the fact that distributed fragments will be involved, and need to be combined
  - User unaware of existence of fragments
  - Allows to fragment a relation horizontally (create a subset of tuples of a relation) or vertically (create a subset of columns of a relation)

Replication Transparency

- Different replicas will be kept consistent and updates to one replica will be propagated transparently to others
  - User should be unaware of existence of copies of data
  - It allows to store copies of data at multiple sites. This is done to minimize access time to the required data.

Distributed Query Processing



Decomposition

- query first analysed for correctness (syntax, etc.)
- **starts with a high-level query and transforms into a query graph of low-level operations (algebraic expressions), which satisfies the query**.

Data localization

- transformation of query into a fragment query
- **determines which fragments are involved in the query and thereby transforms the distributed query into a fragment query**.

Global query optimization

- cost model is used to evaluate different global strategies
- **generates a distributed execution plan so that least amount of data transfer occurs across the sites**.

Local query optimization

- optimal strategy for local execution
  - **Optimization of fragment query on each node using local catalog data (statistics), index structures and subsequently Cost-based selection of locally optimal plan – Similar to CDBMS**

Distributed Query Processing: Challenges

- **Fragmentation and replication**
- **Query optimisation and execution**
- **Transaction management**
- **Security and privacy**
- **Consistency and concurrency**
- **Integration and interoperability**
- **Quality and provenance**
- **Recovery**

Transactions

- *Transactions* are a fundamental concept of all database systems
  - Collection of database operations, executed as a logical unit
  - atomic process that is either performed into completion entirely or is not performed at all

- Eg: update A

  Read A=100

  A=A+10

  Write A

Properties of Transactions: ACID

- Atomic
  - The transaction happens as a single indivisible action. Everything succeeds or else the entire transaction is rolled back. Others do not see intermediate results.
- Consistent
  - A transaction cannot leave the database in an inconsistent state.
- Isolated (Serializable)
  - Transactions cannot interfere with each other. If transactions run at the same time, the final result must be the same as if they executed in some serial order.
- Durable
  - Once a transaction commits, the results are made permanent. No failures after a commit will cause the results to revert.

Commit Protocol

- Commit protocol is used to ensure atomicity across sites
  - a transaction which executes at multiple sites must either be committed at all the sites or aborted at all the sites.
  - not acceptable to have a transaction committed at one site and aborted at another
- Commit Protocols:
  - The **two-phase commit (2 PC)** protocol is widely used
  - The **three-phase commit (3 PC)** protocol is more complicated and more expensive but avoids some drawbacks of the two-phase commit protocol

Two-phase commit (2 PC) Protocol

- Implements transaction atomicity - transaction is either fully executed or not executed at all, and there will be no partial execution

- Each node needs to know if other nodes successfully stored the data or if they failed.
- A mechanism to ensure that all nodes either commit (commit) the transaction or roll back (rollback) the transaction

- Phase 1: **Voting Phase**- preparation
- Phase 2: **Commit Phase**-commit/rollback

Two-phase Commit: Phase 1(Voting phase)

Coordinator

- Write *prepare to commit* to log
- Send *prepare to commit* message
- WAIT state - Wait for all participants to respond or until timeout period reached.

→

←

Participants

- Work on transaction
- Wait for message from coordinator
- Receive the *prepare* message
- When ready, write *agree to commit* or *abort* to the log
- Send *agree to commit* or *abort* to the coordinator

Two-phase Commit: Phase 2(Commit phase)

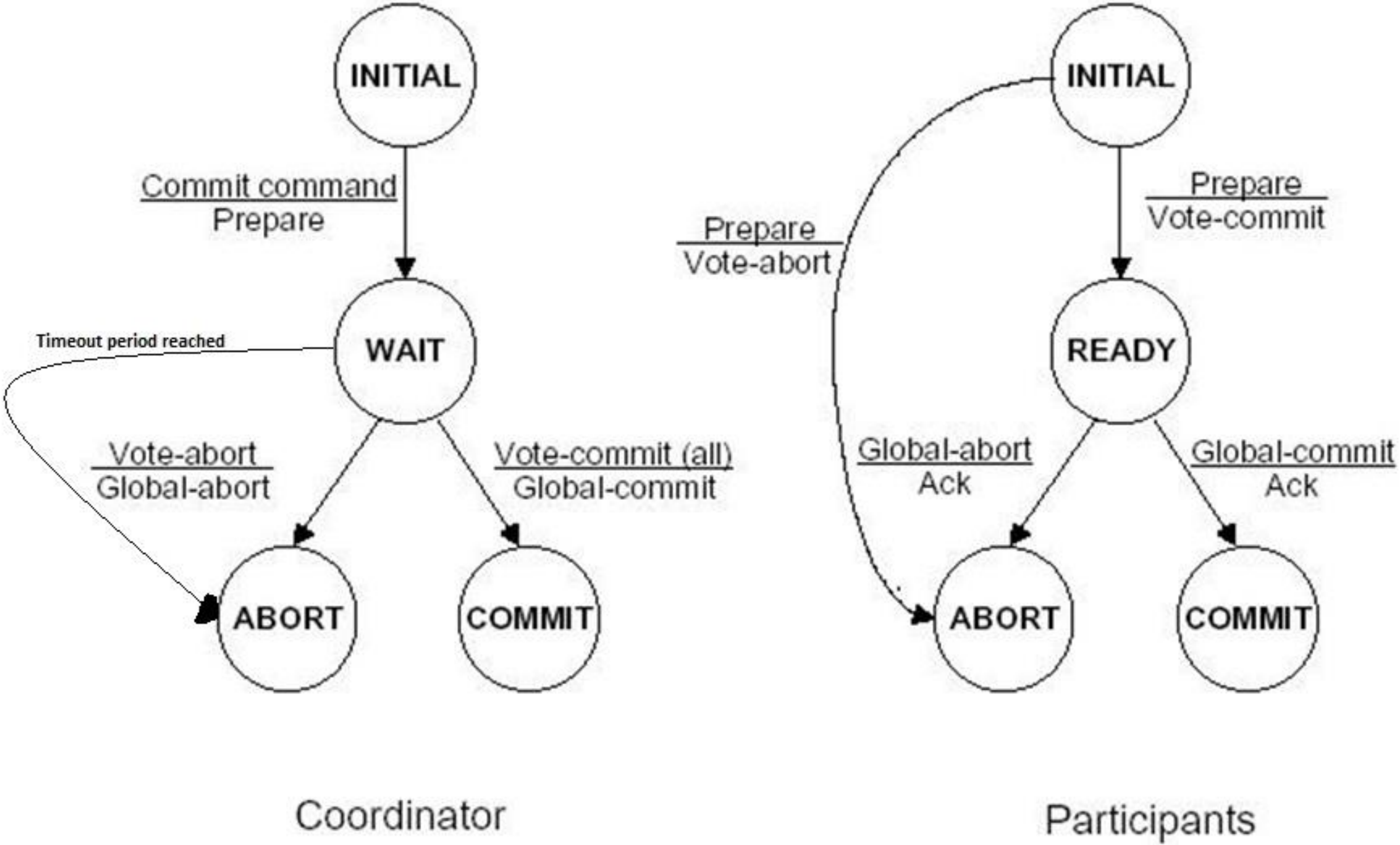
Coordinator

- Write *commit* or *abort* to log
- Send *commit* or *abort*
- Wait for all participants to respond.
- Write log!

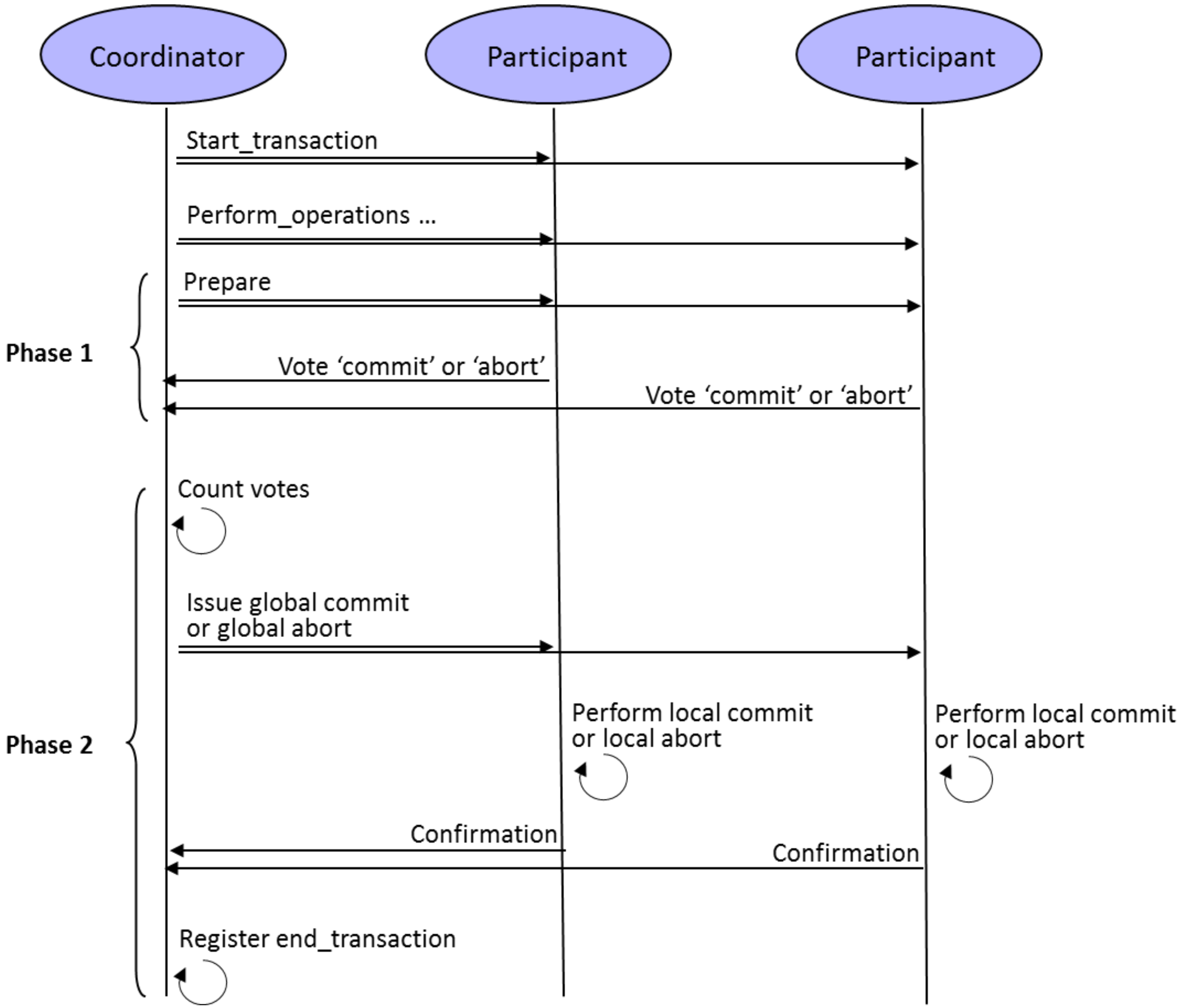
Participants

- Wait for *commit/abort* message
- Receive *commit* or *abort*
- If a *commit* was received, write "*commit*" to the log, release all locks, update databases.
- If an *abort* was received, undo all Changes
- Send *done* message

Two-Phase Commit Protocol (2PC)



2PC Protocol



Two Phase Commit – Participant Failures

- Participant fails in Phase 1
  - Coordinator will timeout and will make an abort decision i.e. will send an abort message to all participants.

- Participant fails in Phase 2
  - Participant has sent RDY message and then fails.
  - Coordinator can make decision to either a commit or abort.
  - When participant recovers, needs to check with coordinator on the decision and complete transaction.
    Coordinator or other participants will not be blocked.
    Coordinator will write log when participant responds.

Two Phase Commit – Coordinator Failures

- **Coordinator fails in Phase 1**
  Coordinator has sent PREPARE message and after that has failed.
  Has not taken a decision on transaction.
  Participant will elect a new coordinator and restart COMMIT Protocol. (Assumption: all participants are live)

Two Phase Commit – Coordinator Failures

- **Coordinator fails in Phase 2**
  Means coordinator has made a decision to either COMMIT or ABORT.

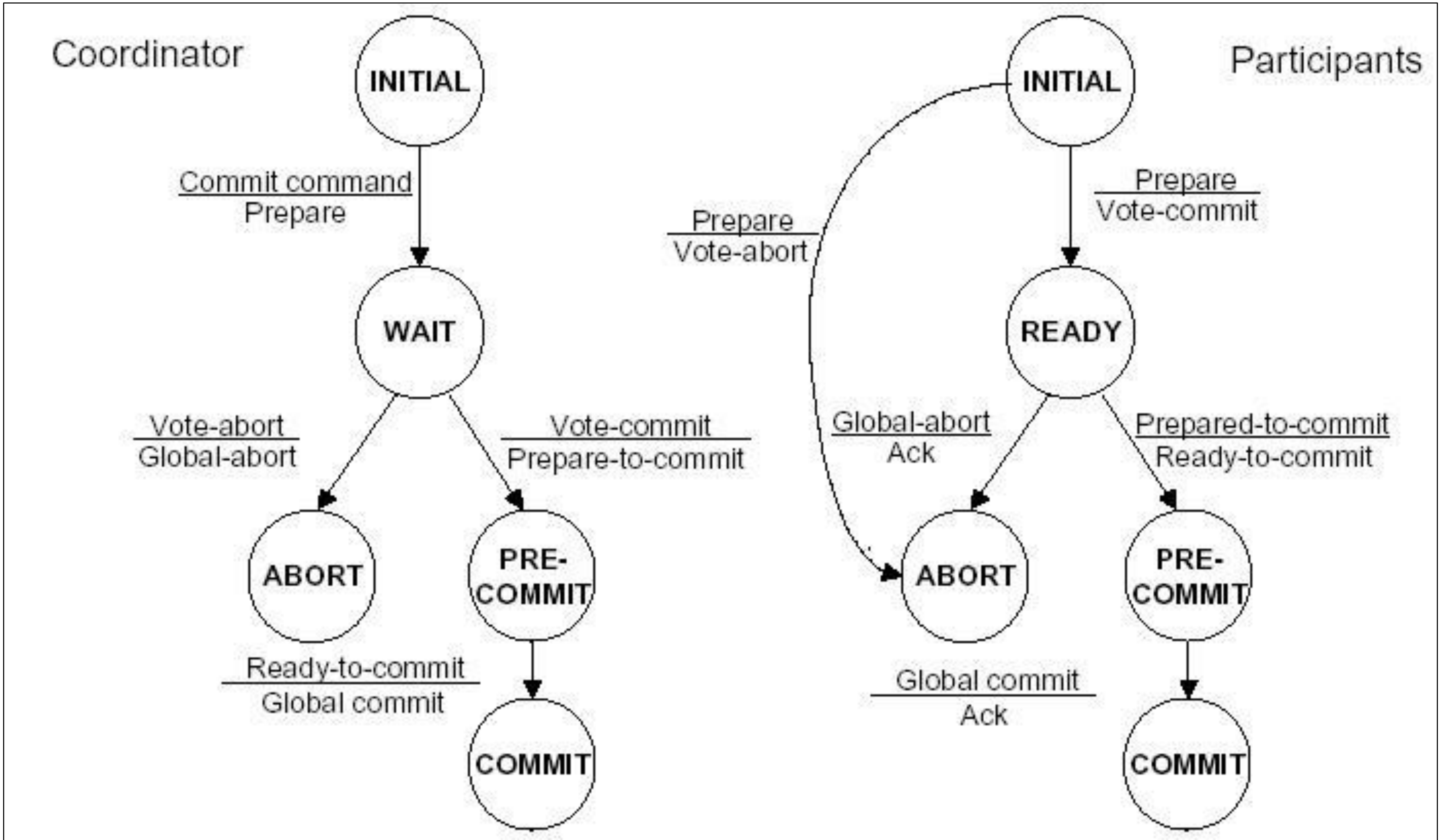
  Scenario 1: If at least one of live participants had received the decision from coordinator. Same decision can be conveyed to all the other participants.
  This is NON-BLOCKING, means nobody is blocked. (not waiting for someone to recover)

  Scenario 2: None of live participants know about the decision of coordinator, i.e. decision is unknown.
  i) All Participants are live, can elect a new coordinator and restart 2 phase commit protocol.
  ii) Coordinator and participant failure.
  In this case all live participants are blocked until coordinator recovers

Three Phase Commit Protocol

- Split the second phase of 2PC into two parts:
- Phase 2. "*Precommit*" phase
  - Send *Prepare to commit* message to all participants when it received a *READY* from all participants in phase 1
  - Participants enter the prepare to commit and 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
- Phase 3. "*Commit*" phase (same as in 2PC)
  - If coordinator gets ACKs for all "*prepare to commit*" messages
  - It will send a *commit* message
  - Else it will abort – send an *abort message* to all participants

3PC Protocol



Some helpful links/resources for further learning

- https://medium.com/the-modern-scientist/distributed-system-vs-centralized-system-d2ad232ac259
- Özsu, M. Tamer, and Patrick Valduriez. "Distributed and parallel database systems." *ACM Computing Surveys (CSUR)* 28.1 (1996): 125-128.
- Ceri, Stefano, Mauro Negri, and Giuseppe Pelagatti. "Horizontal data partitioning in database design." *Proceedings of the 1982 ACM SIGMOD international conference on Management of data*. 1982.
- Navathe, Shamkant, et al. "Vertical partitioning algorithms for database design." *ACM Transactions on Database Systems (TODS)* 9.4 (1984): 680-710.
- Traiger, Irving L., et al. "Transactions and consistency in distributed database systems." *ACM Transactions on Database Systems (TODS)* 7.3 (1982): 323-342.