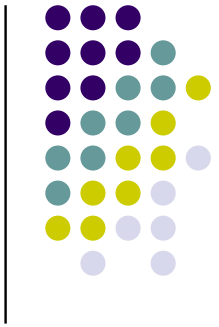
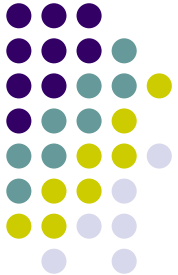


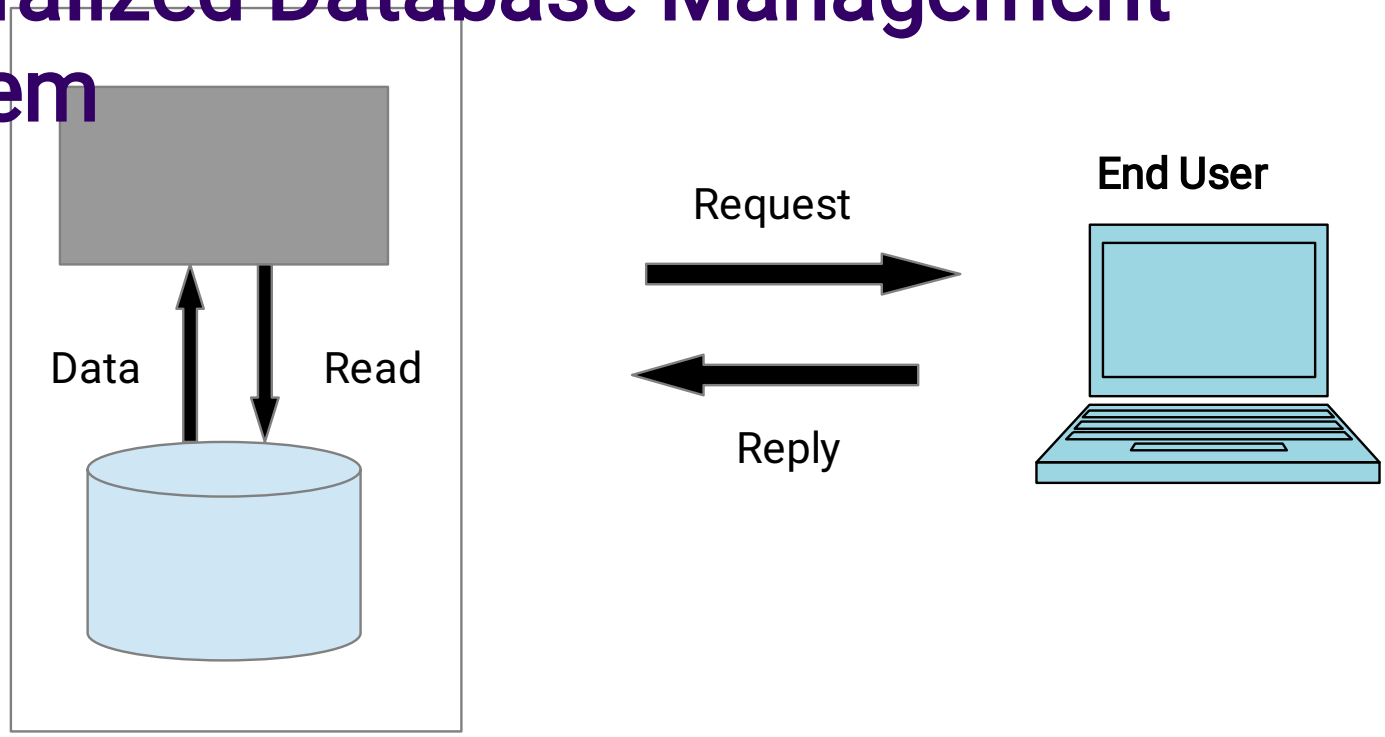
UNIT - 3

DDBMS

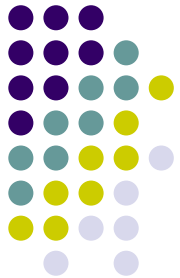


- A **distributed database management system** governs the storage and processing of logically related data over interconnected computer system in which both data and processing function are distributed among several sites.
- **Use** of centralize require that corporate data be stored in a single central site, usually a mainframe computer. But it has a slow response time.

Centralized Database Management System

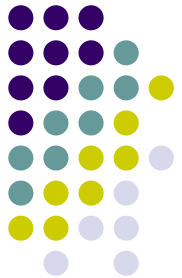


Requirement for DDBMS



- Business operations become more decentralized geographically.
- Competition increased as global level.
- Customer demands & market needs favored a decentralized management style.
- Corporation adopted LAN as basis for their solutions.
- As large business units restructured two database requirements became obvious.
  - Rapid and hoc data access became crucial in the quick-response decision-making environment.
  - Decentralization of management structure having multiple-access and multiple-location database is a necessity.

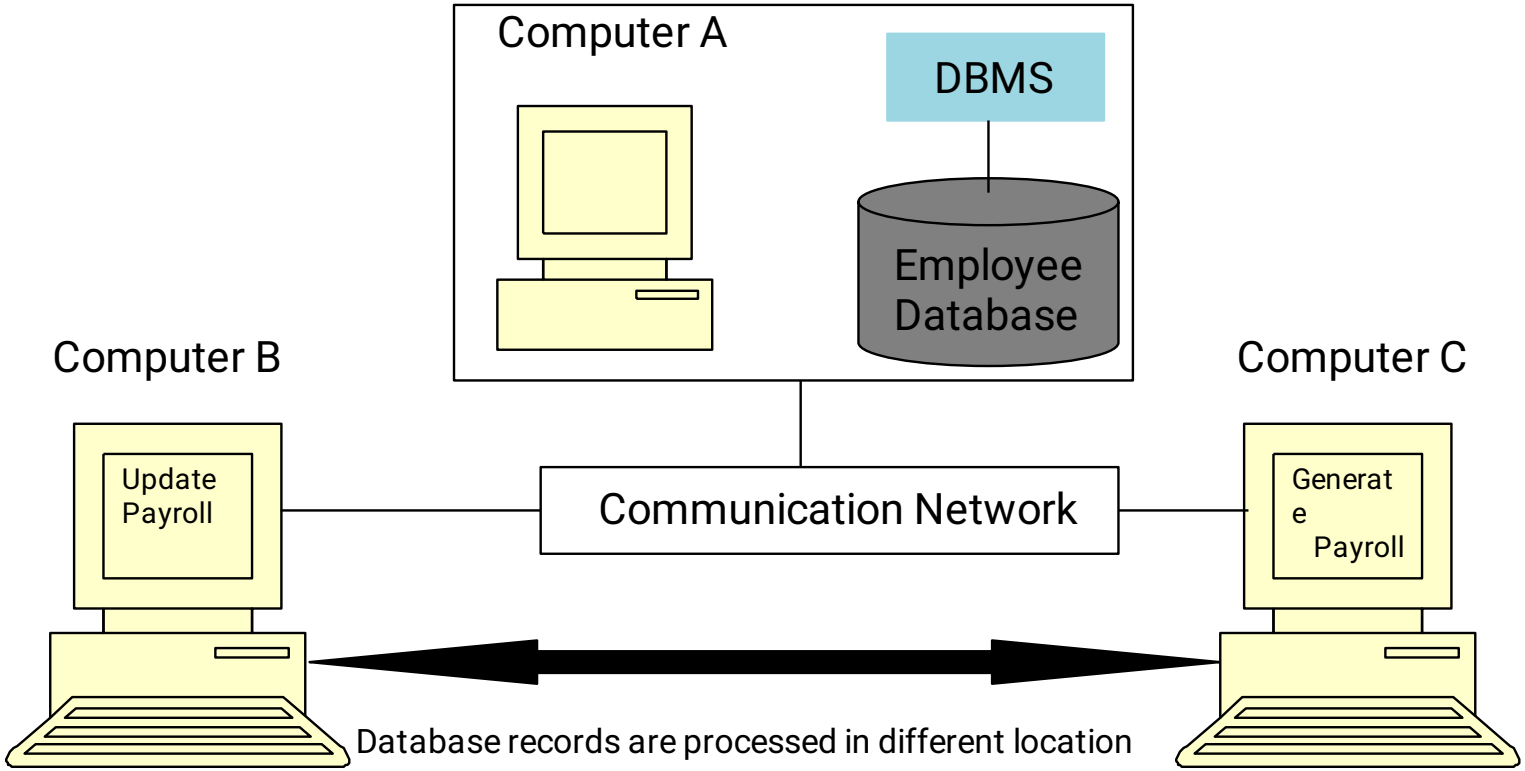
Requirement for DDBMS



- Centralized database is subject to problem such as:
  - Performance degrading due to a growing number of remote location over greater distance.
  - High cost associated with maintaining and operating large central database system.
  - Reliability problems created by dependence on a central site.

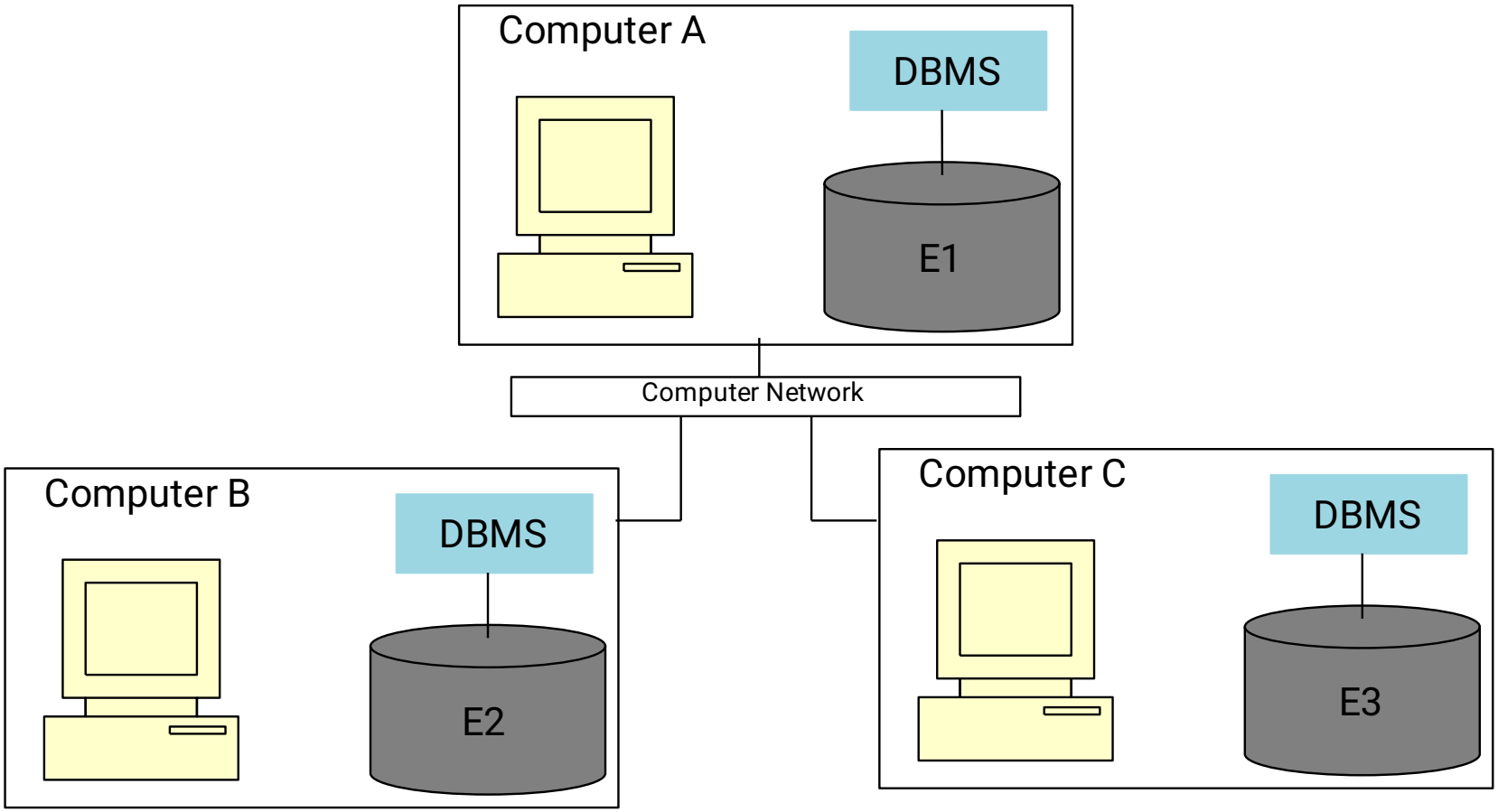
Distributed Processing and Distributed Database.

- A distributed processing is a logical processing is shared among two or more physically independent sites that are connected through a network.

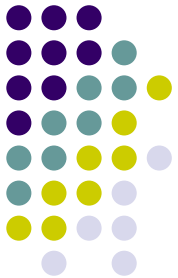


Distributed Processing and Distributed Database.

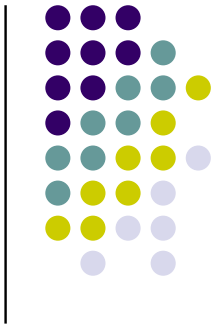
- A distributed Database stores a logically related database over two or more physically independent sites.



Distributed Processing and Distributed Database.

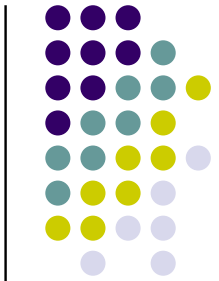


- Consider the following points.
  - Distributed processing does not require a distributed database, but a Distributed database requires distributed processing.
  - Distributed processing may be on a single database located on a single computer.
  - Both distributed database and Distributed Processing requires a network to connect to all components.



Levels of Distributing Data and Processes

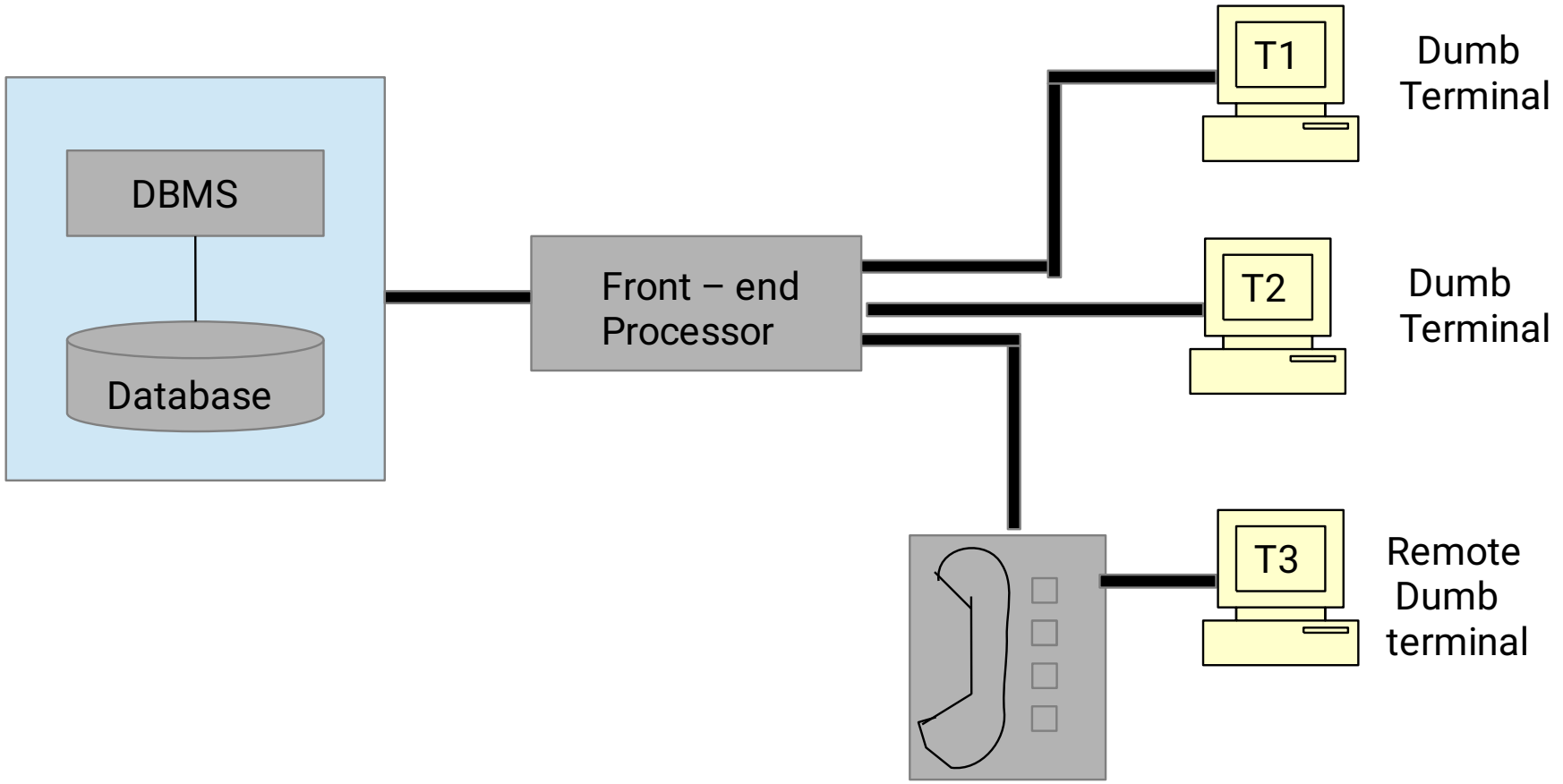
Single-Site processing, single-site data (Centralized)



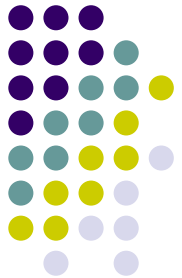
- In a single –site processing, single-site data, all processing is done on a single CPU or host computer and all data are stored on the stored on the host computer's local disk. Processing can not be done on the end user's side of the system.

- The DBMS is located on the host computer , which is accessed by dumb terminal connected to it.

Single-Site processing, single-site data (Centralized)

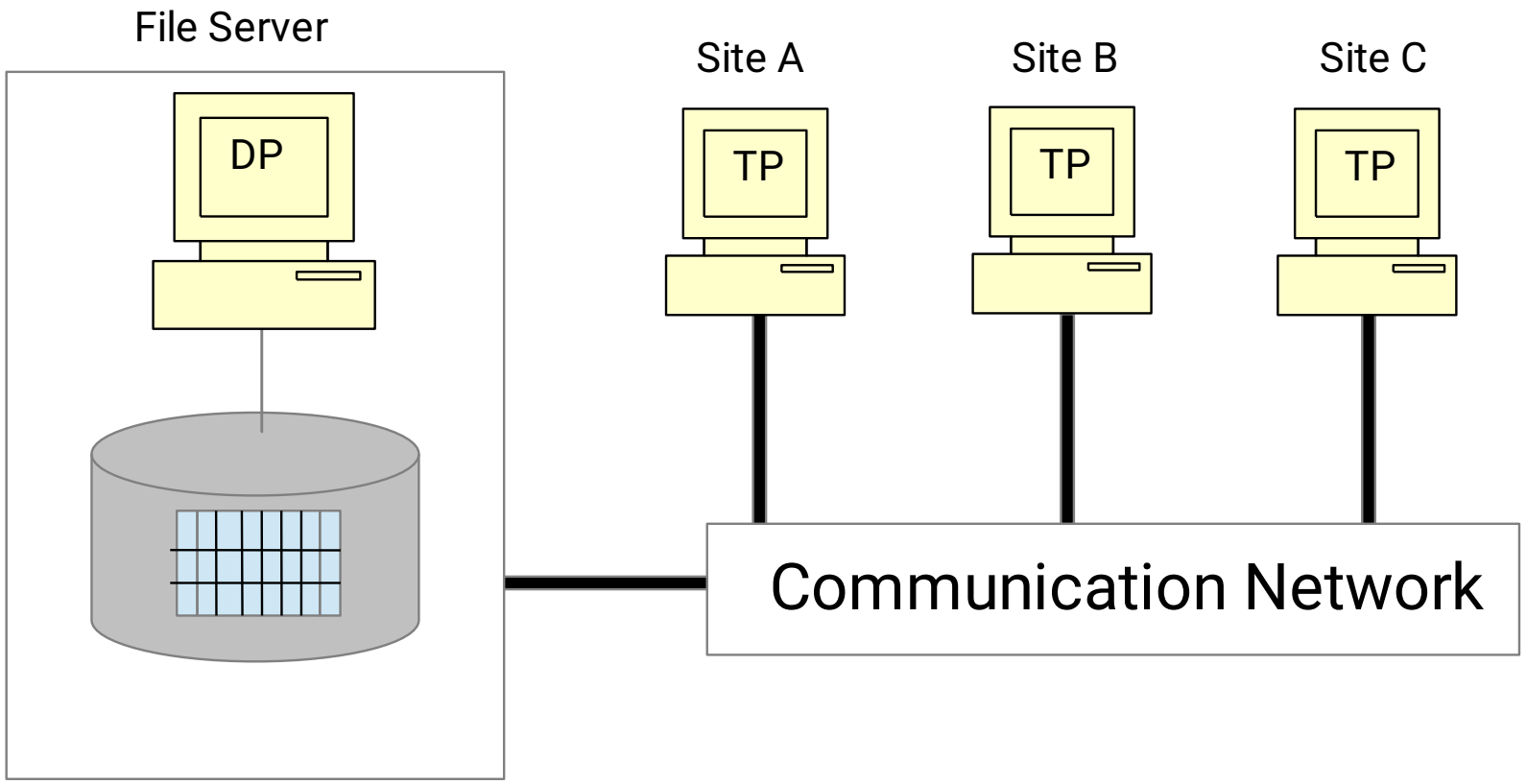


Multiple-site processing, single-site Data

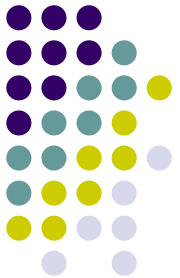


- In Multiple-site processing, single-site data , multiple processes run on different computers sharing single data repository.
- Typically, MPSD requires a network file server running conventional applications that are accessed through a LAN.
  - The TP acts only redirector to route.
  - File Server that act as a only hard disk.
  - The end user must make a direct reference to the file server.
  - All data section, search and update function take place at a TP so require to transfer all data through the network.

Multiple-site processing, single-site Data

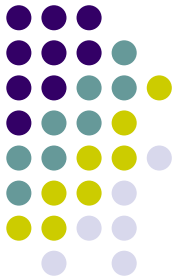


Multiple-site processing multiple-site Data



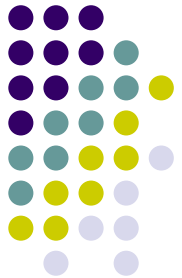
- This is a fully distributed DBMS with support for multiple data processor and transaction processing at multiple site.
- DDBMS are classified as a
  - Heterogeneous DDBMS - Integrate one type of centralized DBMS over a network.
  - Homogeneous DDBMS - Integrate different types of centralized DBMS over a network.
- Limitations of DDBMS
  - Remote access is Read-only
  - Restriction on number of remote tables.
  - Restriction on number of distinct database.
  - Restriction on database models.

Distributed Database Transparency Features



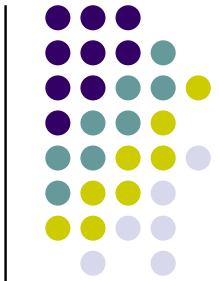
- Transparency features indicate user believes that (s)he is working with centralize DBMS , all complexity of the distributed database are hidden or transparent.
- DDBMS transparency features are :
  - Distribution Transparency : Which allow a distributed database to be treated as a single logical database. User is not aware of
    - Data Partition
    - Data replication at several sitr.
    - Data location.

Distributed Database Transparency Features



- Transaction Transparency : which allow a transaction to update data at several network sites. This ensure that either entire transaction completed or aborted.
- Failure transparency : which ensures that the system will continue to operate in the event of a node failures by redirecting another node.
- Performance transparency : which allows a system to perform as if it were a centralized DBMS. System not suffer from Performance degrading due to network.
- Heterogeneity transparency : which allows integration of several different local DBMS under a common , or global schema.

Distribution Transparency



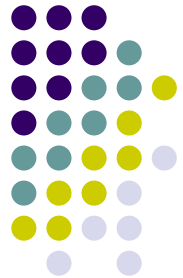
- Distribution transparency allows a physically dispersed database to be managed as though it were a centralized database.
- Three levels transparency are there:
  - Fragmentation transparency : is the highest level transparency. The end user does not need to know that the database partitioned.
  - Location transparency : exists when the end user must specify the database fragment names but does not need to specify where those fragment are located.
  - Local mapping transparency : exists when the end user must specify both fragment names and their locations.

Distribution Transparency

- **Case 1: Fragmentation Transparency**
  - SELECT * FROM employee
    WHERE emp_dob < ’01-jan-1960’;

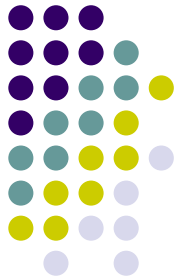
- **Case 2: Location Transparency**
  - SELECT * FROM e1
    WHERE emp_dob < ’01-jan-1960’
  - UNION
    SELECT * FROM e2
    WHERE emp_dob < ’01-jan-1960’
  - UNION
    SELECT * FROM e3
    WHERE emp_dob < ’01-jan-1960’;

Distribution Transparency



- Case 3 : Local Mapping Transparency
  - SELECT * FROM e1 NODE NY
    WHERE emp_dob < ’01-jan-1960’
  - UNION
    SELECT * FROM e2 NODE ATL
    WHERE emp_dob < ’01-jan-1960’
  - UNION
    SELECT * FROM e3 NODE MIA
    WHERE emp_dob < ’01-jan-1960’;

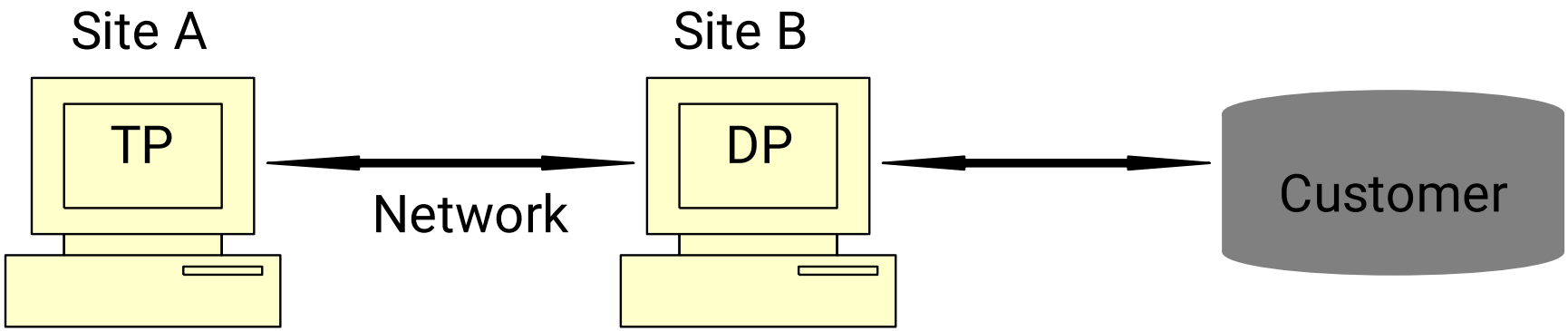
Transaction Transparency



- Distributed Request and Distributed Transparency.
- Distributed Concurrency control.
- Two – phase commit Protocol.

Distributed Request and Distributed Transparency.

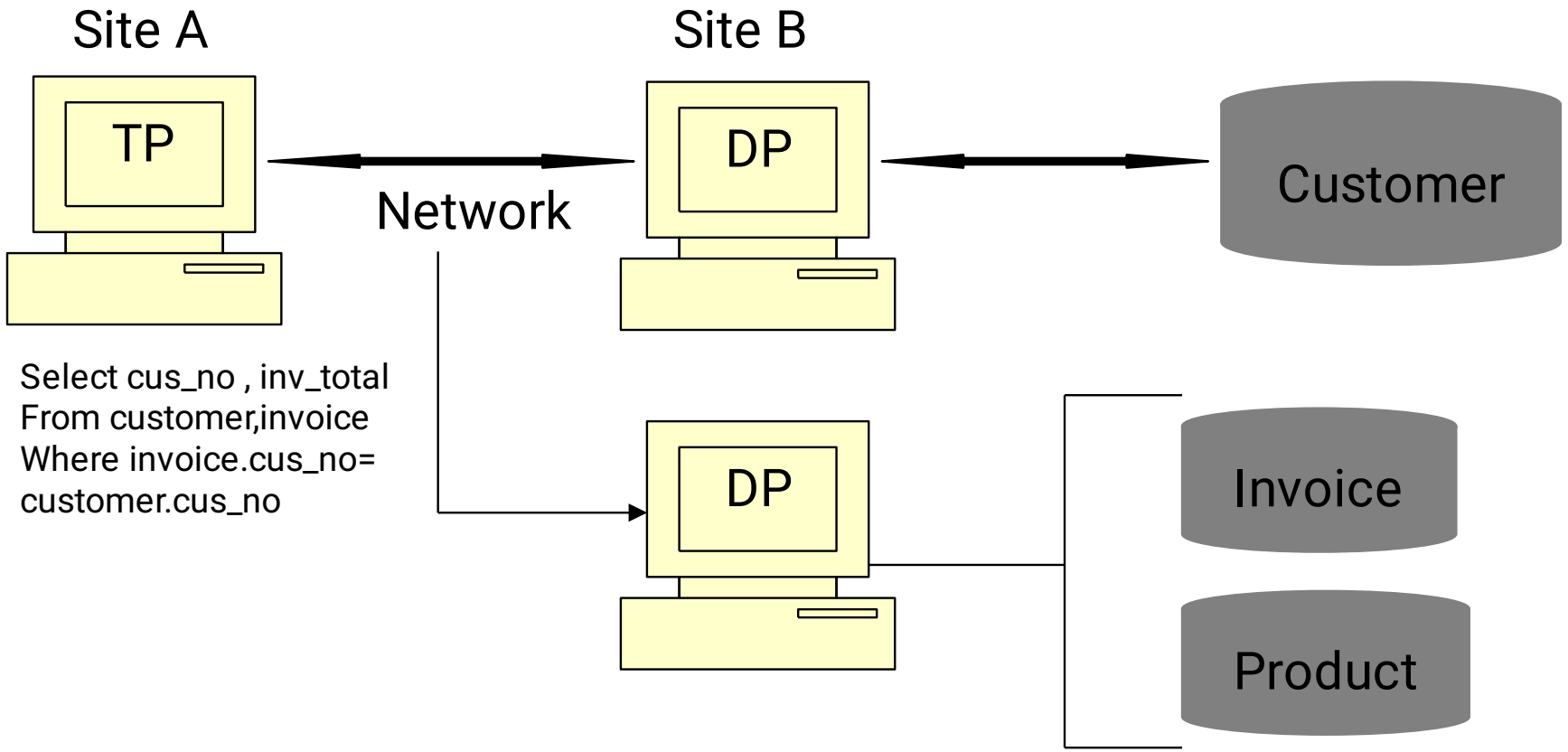
- The Remote request allow single SQL statements access the data that are to be processed by a single remote database processor.



Select * from customer
Where cus_state='AL';

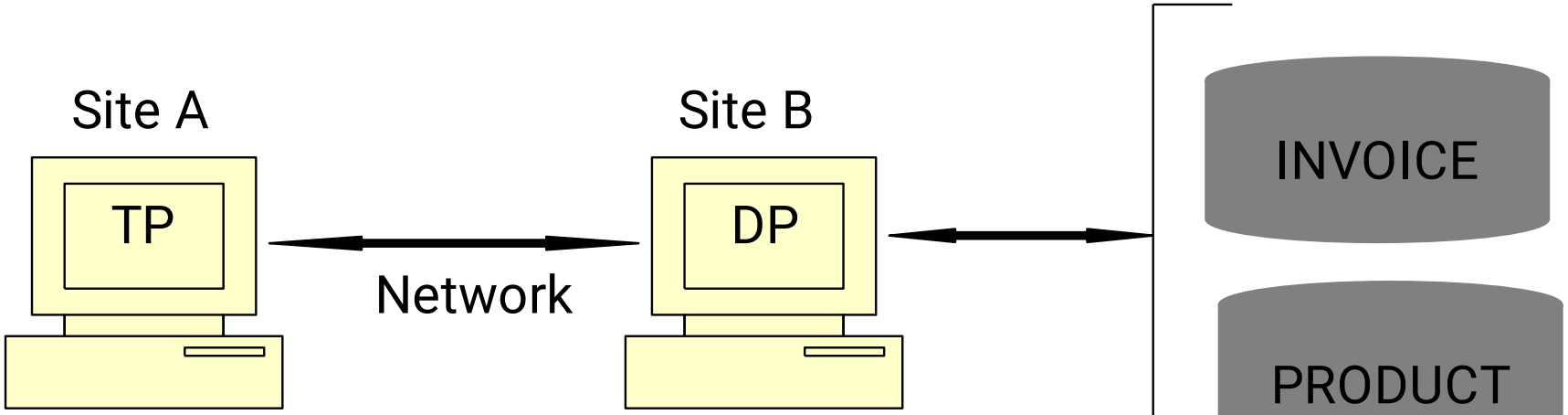
Distributed Request and Distributed Transparency.

- A distributed request allows a single SQL statement reference data located at several different local or remote DP. This allows
  - Partition a database table to fragment
  - Fragmentation transparency.



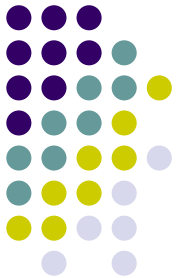
Distributed Request and Distributed Transparency.

- A remote Transaction composed of several request , access data at a single remote site.




```
BEGIN WORK
UPDATE product set qty=10 where prod_num=1234;
Insert into invoice values(100,'15-feb-2006',120);
COMMIT WORK
```

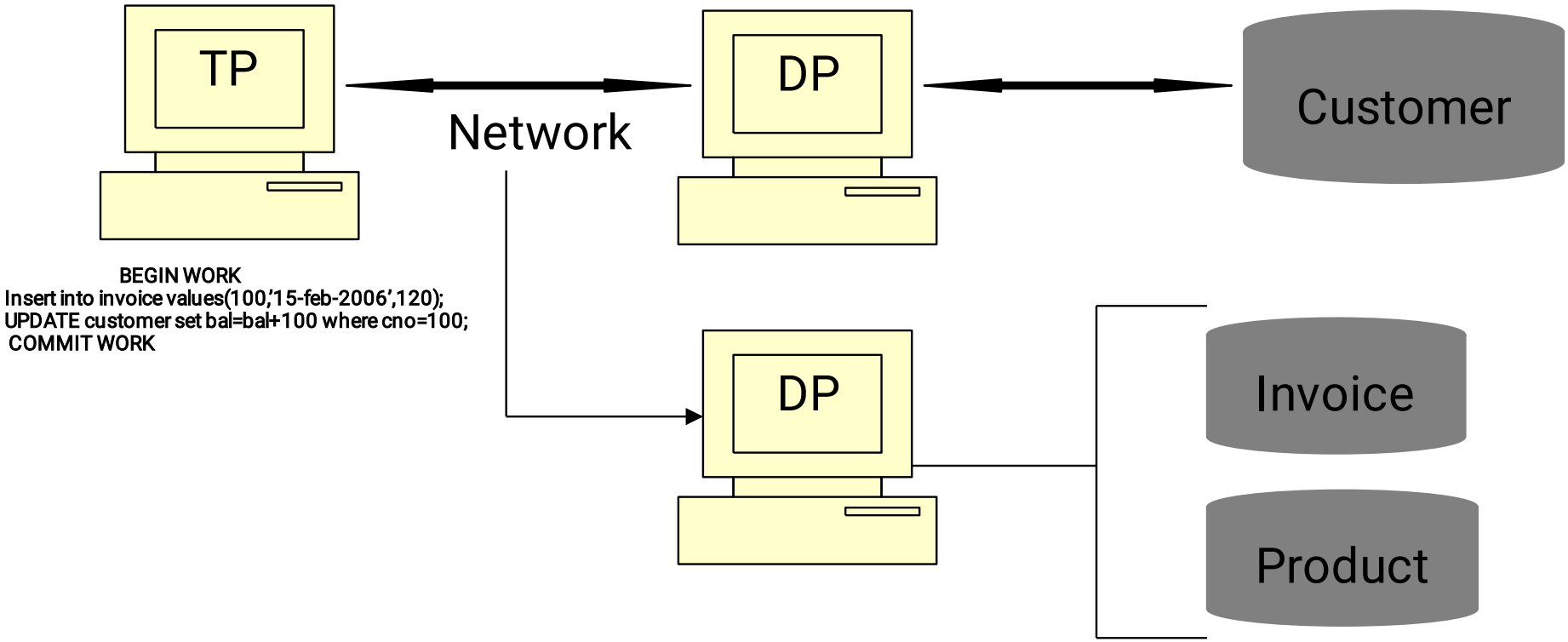
Distributed Request and Distributed Transparency



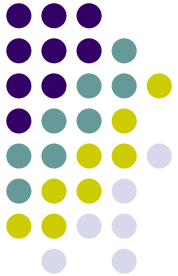
- A remote transaction have a following features:
  - The transaction update PRODUCT and INVOICE table.
  - Remote transaction sent and executed at remote site B.
  - Transaction can reference only one Remote data procedure.
  - SQL statement can reference only one DP at a time.

Distributed Request and Distributed Transparency

- Distributed Transaction allows to reference several different or local or remote DP sites.

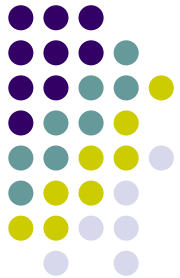


Distributed Request and Distributed Transparency



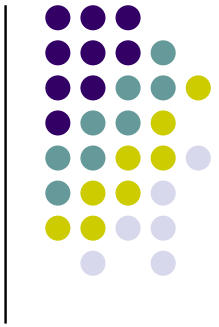
- Distributed Transaction have following features:
  - The transaction references two remote sites B and C.
  - Each request can access only one remote sites at a times.

Distributed Concurrency Control



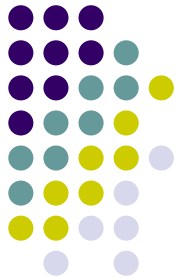
- Concurrency is important in distributed environment because multisite, multiple-process operation are more likely to create data inconsistencies and deadlocked transaction.
- Suppose each transaction was committed by each local DP but one of the DP could not commit the result. Such a scenario yield inconsistence database that can be solved by TWO-PHASE COMMIT protocol.

Two-phase commit protocol



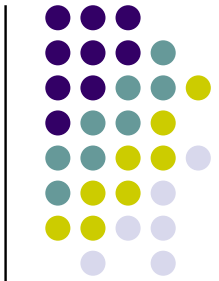
- Each DP maintains its own log. The two-phase commit protocol requires transaction entry log for each DP be written before the database fragment is actually updated. Therefore it requires DO-UNDO-REDO and write-ahead protocol.
- The DO-UNDO-REDO protocol defines three types of operation.
  - DO perform the operation and records the “before” and “after” value in transaction log.
  - UNDO reserves an operation
  - REDO redoes an operation.
- The write-ahead protocol forces the log entry to be written to permanent storage before the actual operation take place.

Two-phase commit protocol



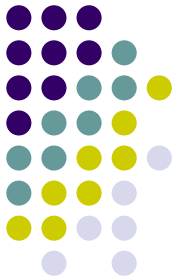
- The two-phase commit protocol defines operation between two types of nodes : The coordinator and one or more subordinator.
- The protocol implement in two phases:
  - **Phase 1: Preparation**
    - 1. The coordinator sends a PREPARE TO COMMIT message to all subordinates.
    - 2. The subordinates receives a message, write transaction log using write-ahead protocol & sends a acknowledgment (YES or NO).
    - 3. The coordinator makes a sure that all nodes are ready to commit , or it abort the action.

Two-phase commit protocol



- **Phase 2 : The Final Commit**
  - 1. The coordinator broadcast a COMMIT messages to all subordinates and wait for reply.
  - Each subordinates receives the COMMIT message, Then update the database using DO protocol.
  - The subordinates reply with COMMITED and NOT COMMITED message to coordinates.

Performance Transparency and Query Optimization



- The objective of query optimization is to minimize the total cost associated with the execution of request.
- The associated with a request are:
  - Access time (I/O) on disk.
  - Communication cost.
  - CPU time cost.