LECTURE 8

DISTRIBUTED DATABASE MANAGEMENT SYSTEMS (DDBMS)

Muhammad Hamiz Mohd Radzi
Faiqah Hafidzah Halim

Content

Objectives

- At the end of this lesson, you should be able to:
  - *Describe Distributed Database (DDB), DDBMS, distributed processing, shared disk, shared memory and shared nothing of parallel DBMS*
  - *Explain the advantages and disadvantages of DDBMS*
  - *Describe type of homogeneous & heterogeneous DDBMS and the Multi Database System (MDBS)*
  - *Explain functions and reference architecture of DDBMS, MDBS and components of DDBMS architecture*

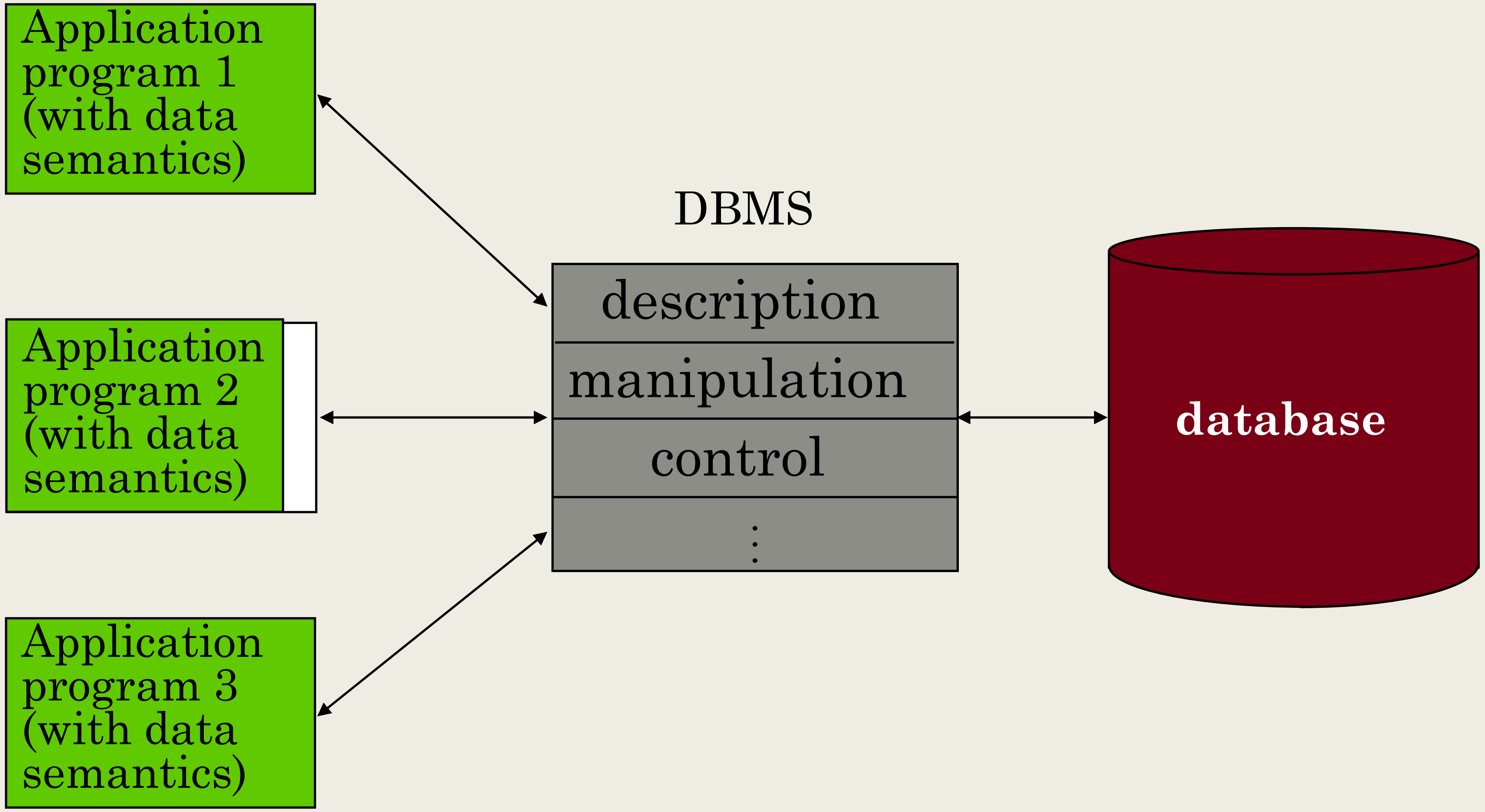
- *Explain the concept of allocation in centralized, fragmented, complete and partial replication of Distributed Relational Database Design (DDD)*

- *Explain the horizontal, vertical, mixed and derived fragmentation together with its correctness rules*

- *Describe the distribution, transaction, performance and DBMS transparencies.*

- *Describe the fragment, location, local mapping and naming in Distribution Transparency.*

DBMS

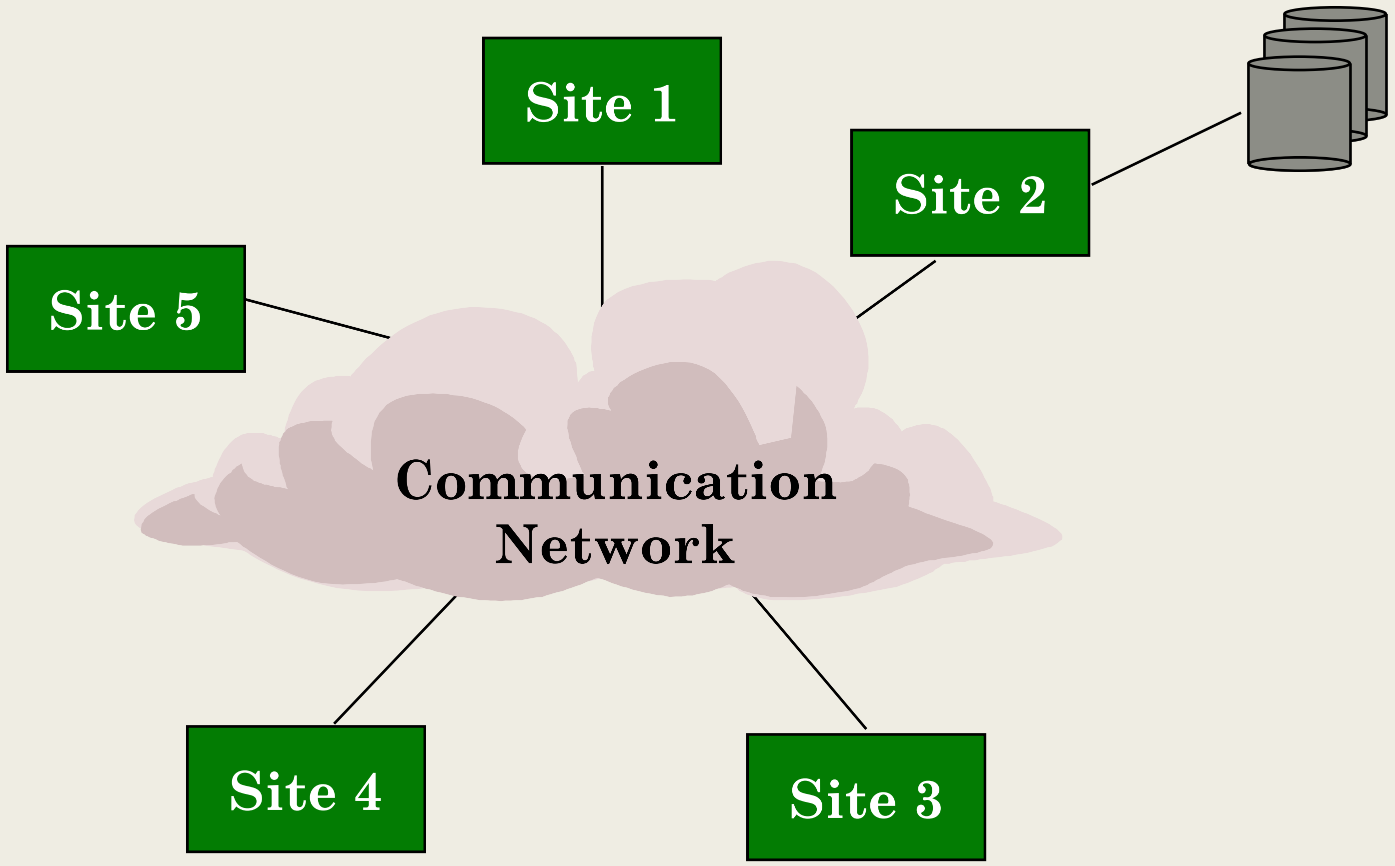

Application program 1 (with data semantics)
Application program 2 (with data semantics)
Application program 3 (with data semantics)
DBMS
description
manipulation
control
⋮
database

DBMS Approach

- DB is located at the server
- Processing is split between server and client
- Less data traffic on the network

Centralized Database (Distributed Processing)

- A database system which resides at one of the nodes of a network of computers.



Problems with Centralized DB

- *Performance degradation as number of remote sites grew*
- *High cost to maintain large centralized DBs*
- *Reliability problems with one, central site*
- *The site with the database can become a bottleneck.*
- *Data availability is not efficient*
- *Possible availability problem: if the site with the database goes down, there can be no data access.*

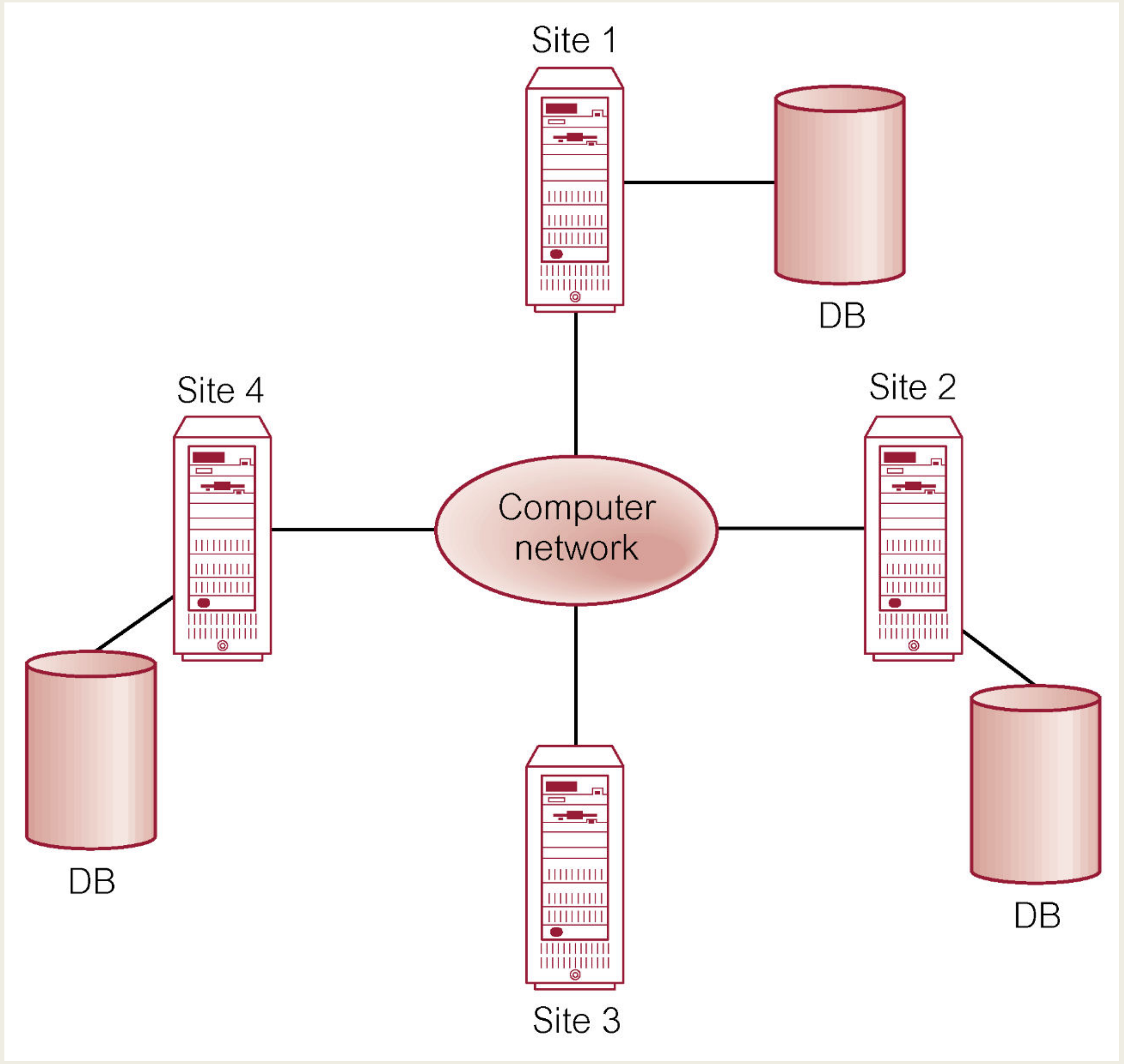
Concept of DDBMS

- Hence, to overcome the problem of centralized DBMS, DDBMS is introduced.

- *Distributed Database: A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network.*

- *Distributed DBMS (DDBMS): Software system that permits the management of the distributed database and makes the distribution transparent to users.*

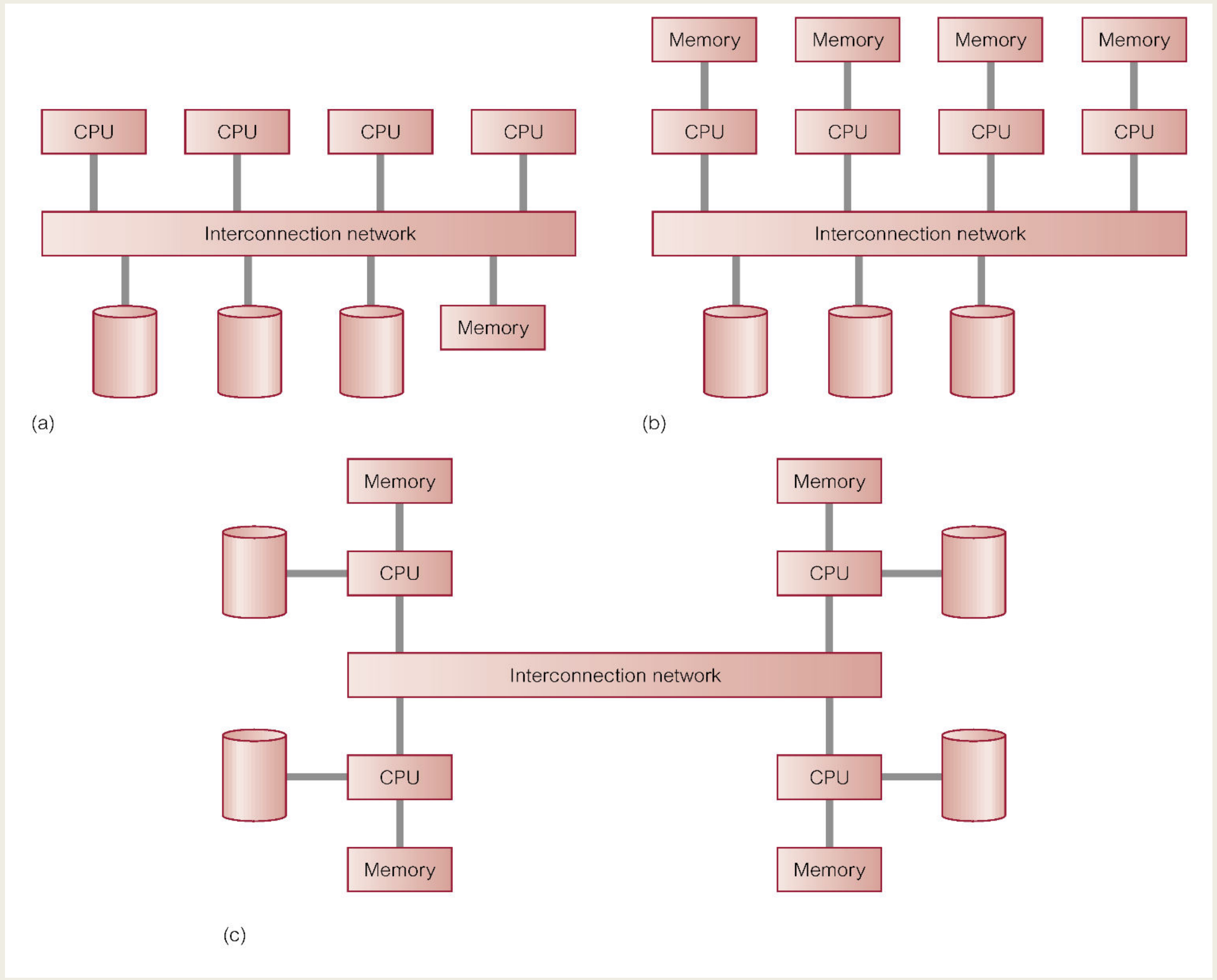
- Collection of logically-related shared data.
- Data split into fragments.
- Fragments may be replicated.
- Fragments/replicas allocated to sites.
- Sites linked by a communications network.
- Data at each site is under control of a DBMS.
- DBMSs handle local applications autonomously.
- Each DBMS participates in at least one global application.


Site 1
DB
Site 4
Site 2
Computer network
DB
DB
Site 3

Parallel DBMS

- A DBMS running across multiple processors and disks designed to execute operations in parallel, whenever possible, to improve performance.

- Main architecture are:
  - *Shared memory*
  - *Shared disk*
  - *Shared nothing*

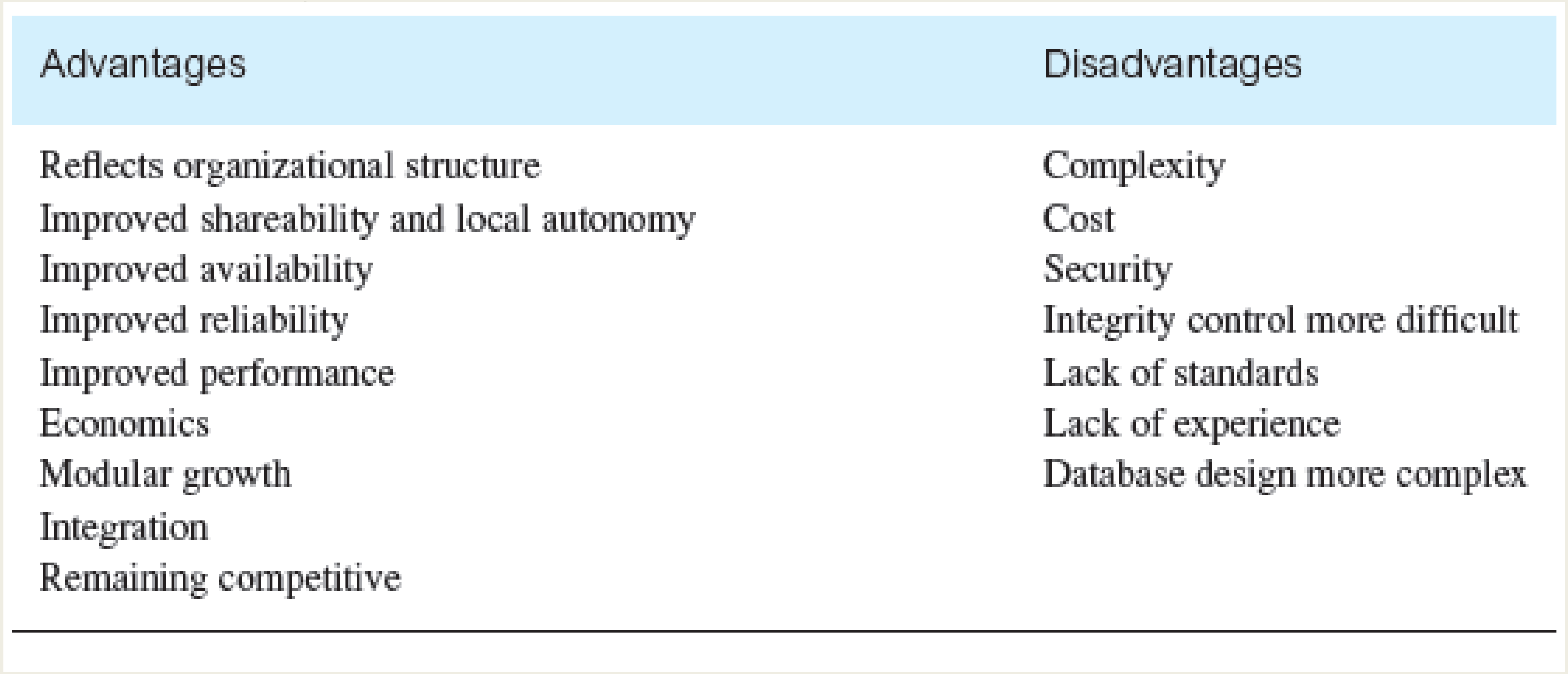


(a) shared memory

(b) shared disk

(c) shared nothing

Advantages & Disadvantages of



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	

Types of DDBMS

- Homogeneous
  - *All sites use same DBMS product.*
  - *Much easier to design and manage.*
  - *Approach provides incremental growth and allows increased performance.*

- **Heterogeneous:**
  - *Sites may run different DBMS products, with possibly different underlying data models.*
  - *Occurs when sites have implemented their own databases and integration is considered later.*
  - *Translations required to allow for:*
  - *Different hardware.*
  - *Different DBMS products.*
  - *Different hardware and different DBMS products.*

Multi Database Systems (MDBS)

- DDBMS in which each site maintains complete autonomy.
- DBMS that resides transparently on top of existing database and file systems and presents a single database to its users.
- Allows users to access and share data without requiring physical database integration.
- Unfederated MDBS (no local users) and federated MDBS.

Functions & Architecture of DDBMS

- Functions: Expect DDBMS to have at least **the functionality** of a DBMS.

- Also to have following functionality:
  - *Extended communication services.*
  - *Extended Data Dictionary.*
  - *Distributed query processing.*
  - *Extended concurrency control.*
  - *Extended recovery services.*

- Global Conceptual Schema (GCS): Logical description of the whole database which contains definitions of entities, relationships, constraints, security, and integrity information.

- Fragmentation schema is a description of how the data is to be logically partitioned.

- The allocation schema is a description of where the data is to be located, taking account of any replication.

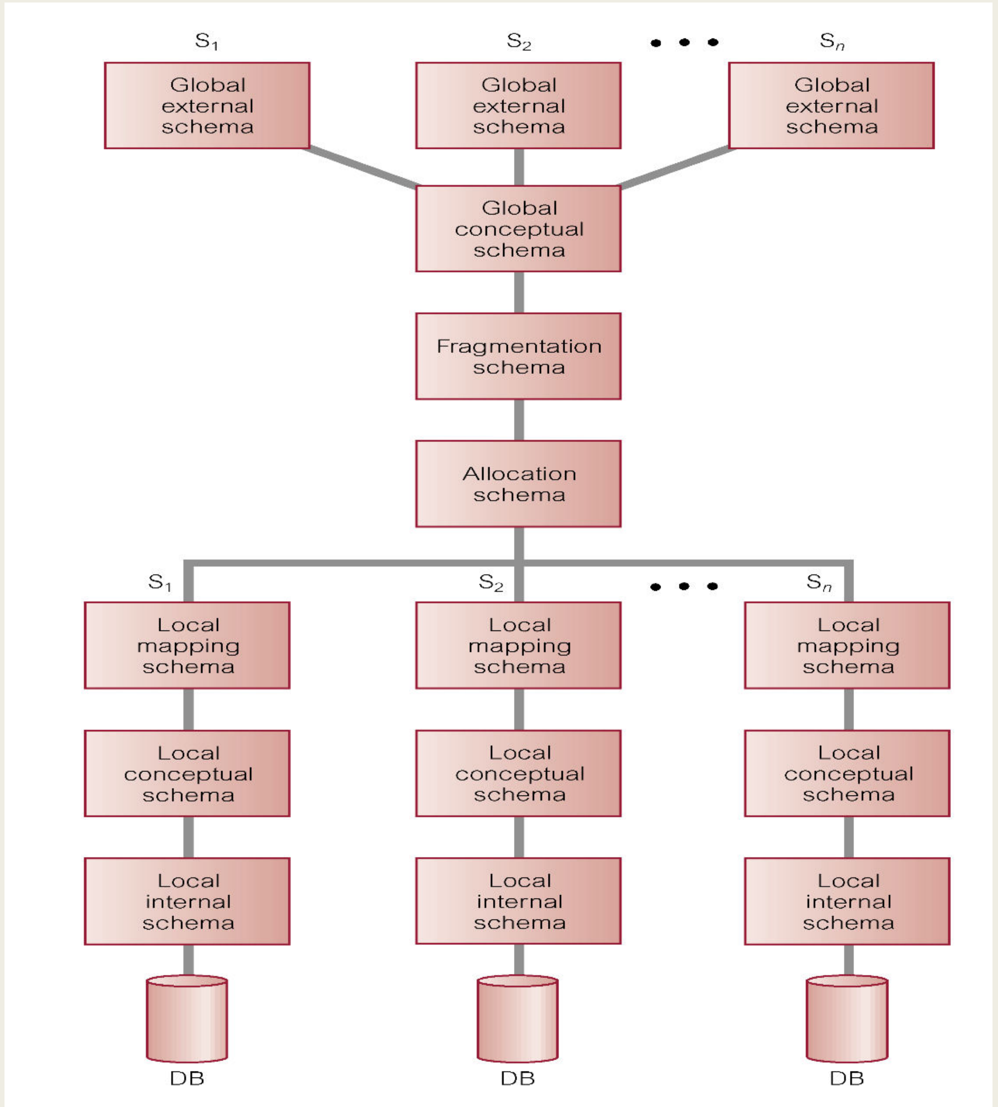
- Local schemas: Each local DBMS has its own set of schemas.

Reference Architecture for DDBMS

- Due to diversity, no accepted architecture equivalent to ANSI/SPARC 3-level architecture.

- A reference architecture consists of:
  - *Set of global external schemas.*
  - *Global conceptual schema (GCS).*
  - *Fragmentation schema and allocation schema.*
  - *Set of schemas for each local DBMS conforming to 3-level ANSI/SPARC.*

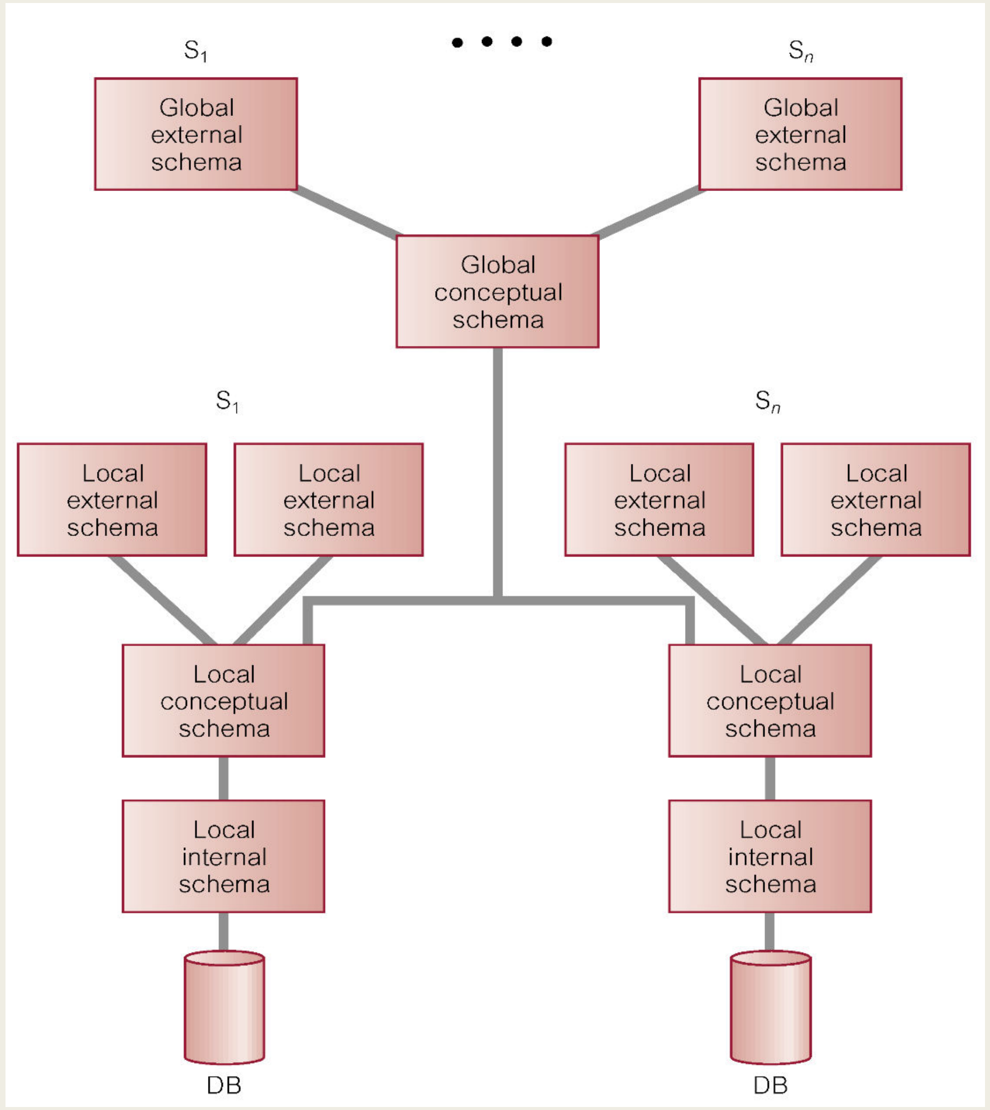
Reference Architecture for DDBMS



Reference Architecture for FMDBS

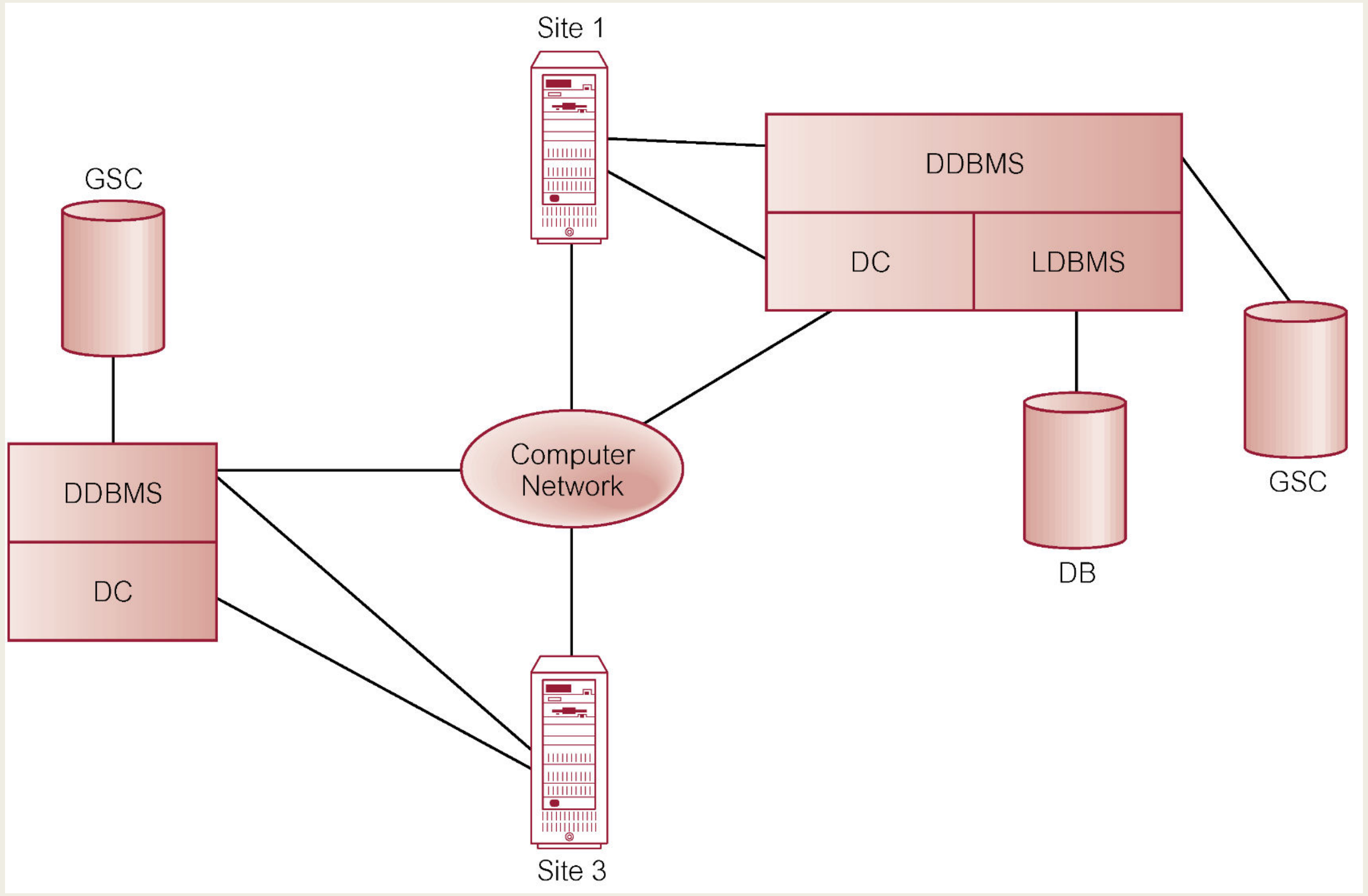
- In DDBMS, GCS is union of all local conceptual schemas.
- In FMDBS, GCS is subset of local conceptual schemas (LCS), consisting of data that each local system agrees to share.
- GCS of *tightly coupled system* involves integration of either parts of LCSs or local external schemas.
- FMDBS with no GCS is called *loosely coupled*.

Reference Architecture for Tightly-Coupled FMDBS



Pearson Education © 2009

Components of DDBMS Architecture



- Global System Catalog (GSC): Holds information such as the fragmentation, replication, and allocation schemas.

- Local DBMS (LDBMS): Controlling the local data at each site that has a database.

- Data Communications (DC): Software that enables all sites to communicate with each other

Distributed Relational Database Design

- Data fragmentation:
  - *How to partition the database into fragments*
- Data replication:
  - *Which fragments to replicate*
- Data allocation:
  - *Where to locate those fragments and replicas*

Fragmentation

- Definition and allocation of fragments carried out strategically to achieve:
  - *Locality of Reference.*
  - *Improved Reliability and Availability.*
  - *Improved Performance.*
  - *Balanced Storage Capacities and Costs.*
  - *Minimal Communication Costs.*
- Involves analyzing most important applications, based on quantitative/qualitative information.

Data Allocation

- <u>Centralized</u>: Consists of single database and DBMS stored at one site with users distributed across the network.
- <u>Partitioned</u>: Database partitioned into disjoint fragments, each fragment assigned to one site.
- <u>Complete Replication</u>: Consists of maintaining complete copy of database at each site.
- <u>Selective Replication</u>: Combination of partitioning, replication, and centralization.

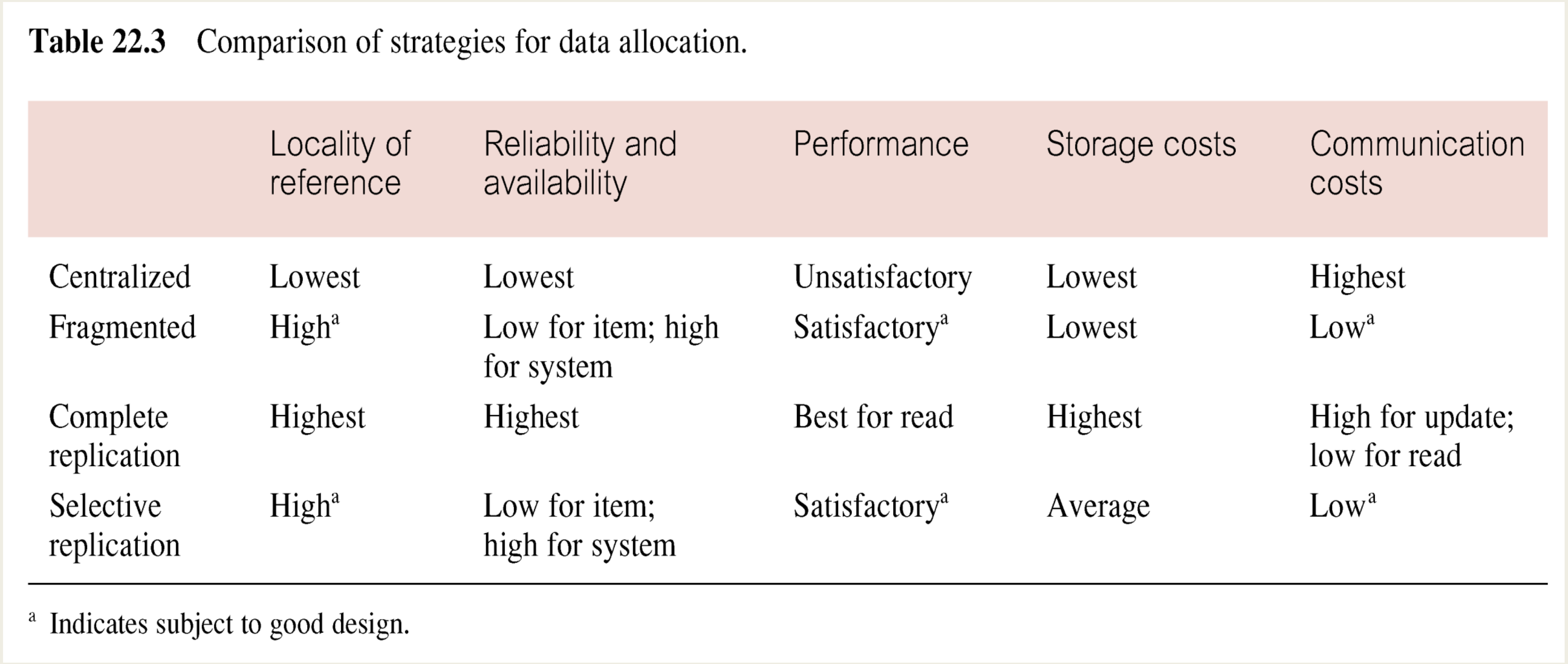


Table 22.3 Comparison of strategies for data allocation.

	Locality of reference	Reliability and availability	Performance	Storage costs	Communication costs
Centralized	Lowest	Lowest	Unsatisfactory	Lowest	Highest
Fragmented	High[a]	Low for item; high for system	Satisfactory[a]	Lowest	Low[a]
Complete replication	Highest	Highest	Best for read	Highest	High for update; low for read
Selective replication	High[a]	Low for item; high for system	Satisfactory[a]	Average	Low[a]

[a] Indicates subject to good design.

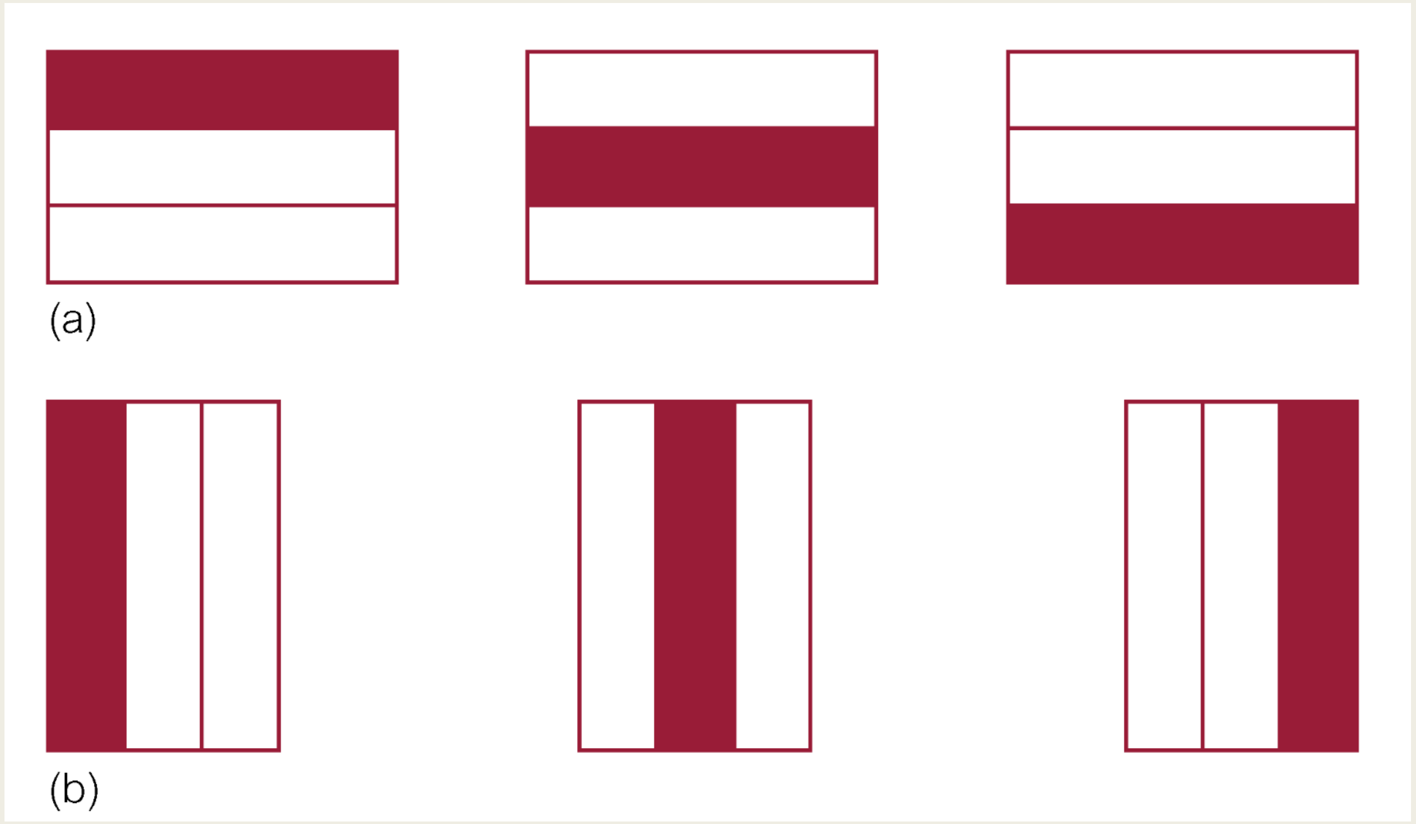
Reasons for Fragmentation

- **Usage:** Applications work with views rather than entire relations.
- **Efficiency:** Data is stored close to where it is most frequently used.
- **Parallelism:** With fragments as unit of distribution, transaction can be divided into several subqueries that operate on fragments.
- **Security:** Data not required by local applications is not stored and so not available to unauthorized users.

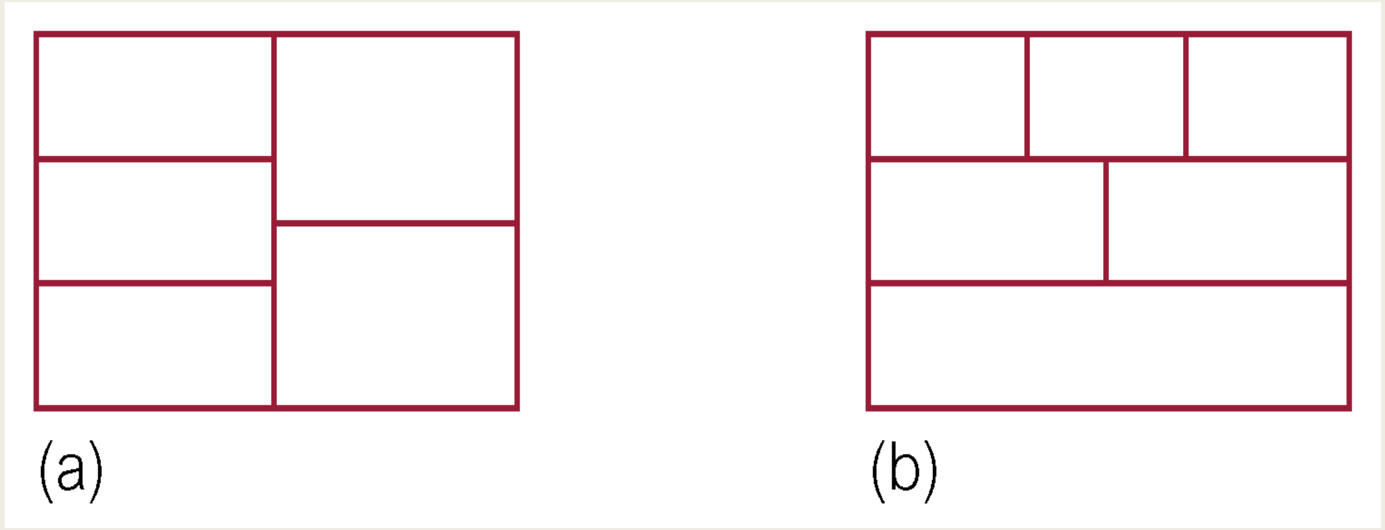
Types of Fragmentation

- Four types of fragmentation:
  - *Horizontal,*
  - *Vertical,*
  - *Mixed,*
  - *Derived.*
- Other possibility is **no fragmentation**:
- If relation is **small and not updated frequently**, may be better not to fragment relation.

Horizontal and Vertical Fragmentation



Mixed Fragmentation



Horizontal Fragmentation

- Consists of a subset of the tuples of a relation.

- Defined using *Selection* operation of relational algebra:

$$\sigma_p(R)$$

- Assuming that there are only two property types, Flat and House, the horizontal fragmentation of PropertyForRent by property type can be obtained as follows:

$$P_1 = \sigma_{type='House'}(PropertyForRent)$$
$$P_2 = \sigma_{type='Flat'}(PropertyForRent)$$

Vertical Fragmentation

- Consists of a subset of attributes of a relation.
- Defined using *Projection* operation of relational algebra:

$$\Pi_{a1, \ldots, an}(R)$$

- For example:

$$S_1 = \Pi_{staffNo,\ position,\ sex,\ DOB,\ salary}(Staff)$$
$$S_2 = \Pi_{staffNo,\ fName,\ lName,\ branchNo}(Staff)$$

- Determined by establishing *affinity* of one attribute to another.

Mixed Fragmentation

- Consists of a horizontal fragment that is vertically fragmented, or a vertical fragment that is horizontally fragmented.

- Defined using *Selection* and *Projection* operations of relational algebra:

$$\sigma_p(\Pi_{a1, \ldots, an}(R)) \qquad or$$
$$\Pi_{a1, \ldots, an}(\sigma_p(R))$$

Derived Horizontal Fragmentation

- A horizontal fragment that is based on horizontal fragmentation of a parent relation.

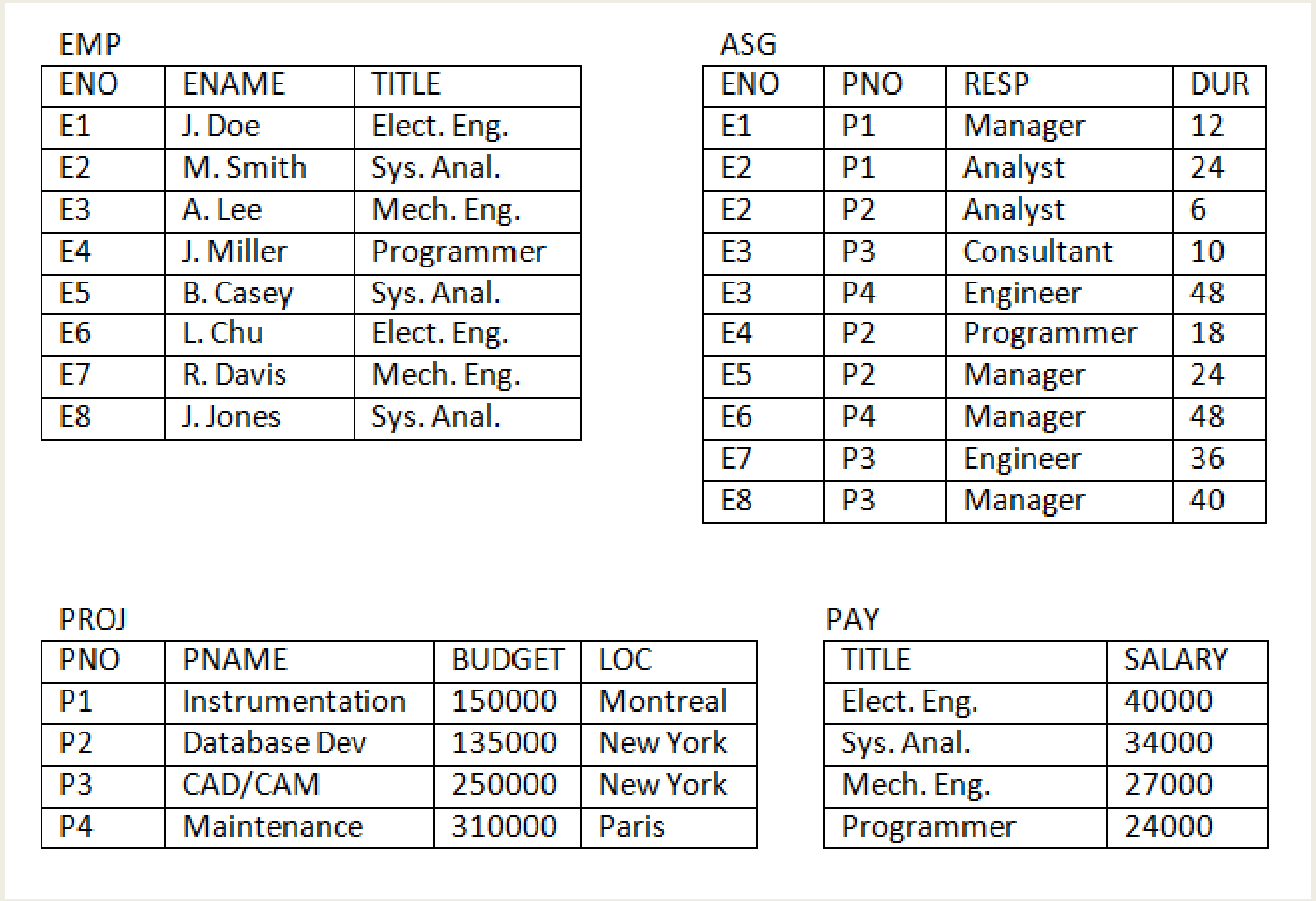
- Ensures that fragments that are frequently joined together are at same site.

- Defined using *Semijoin* operation of relational algebra:

$$R_i = R \rtimes_F S_i, \qquad 1 \leq i \leq w$$

Case study

- Supposed that we have these tables in our database.



EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Sys. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Sys. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Sys. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

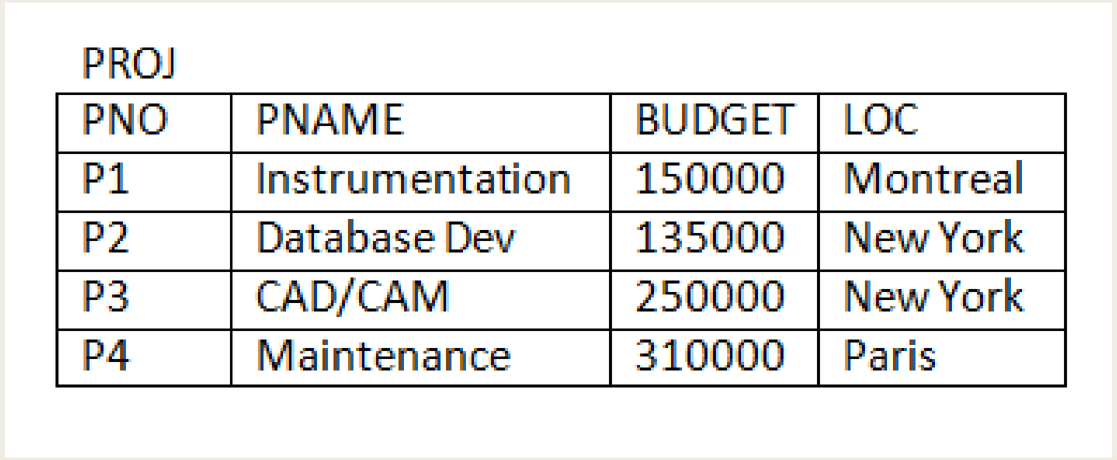
PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Dev	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	SALARY
Elect. Eng.	40000
Sys. Anal.	34000
Mech. Eng.	27000
Programmer	24000

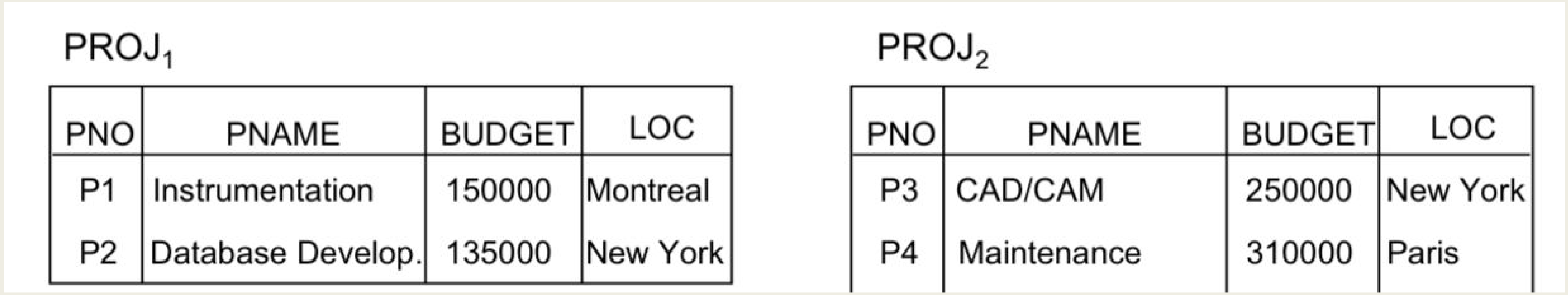
Question 1

- Do a horizontal fragmentation based on:
  - *PROJ1: projects with budget less than $200,000*
  - *PROJ2: projects with budget greater than or equal to $200,000*



PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Dev	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

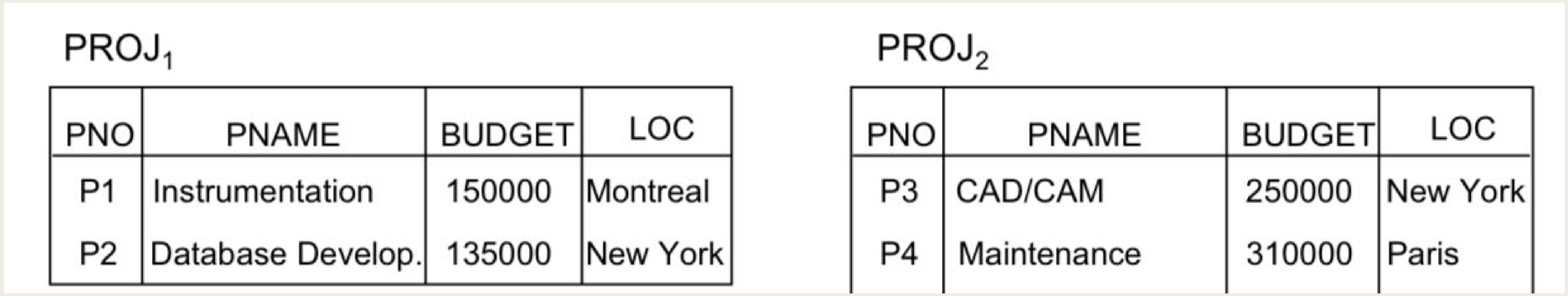


$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris



PROJ$_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

PROJ$_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

By using RA:

$\text{Proj}_1 = \sigma_{\text{BUDGET<200K}} (\text{Proj})$

$\text{Proj}_2 = \sigma_{\text{BUDGET>=200K}} (\text{Proj})$

Reconstruction:

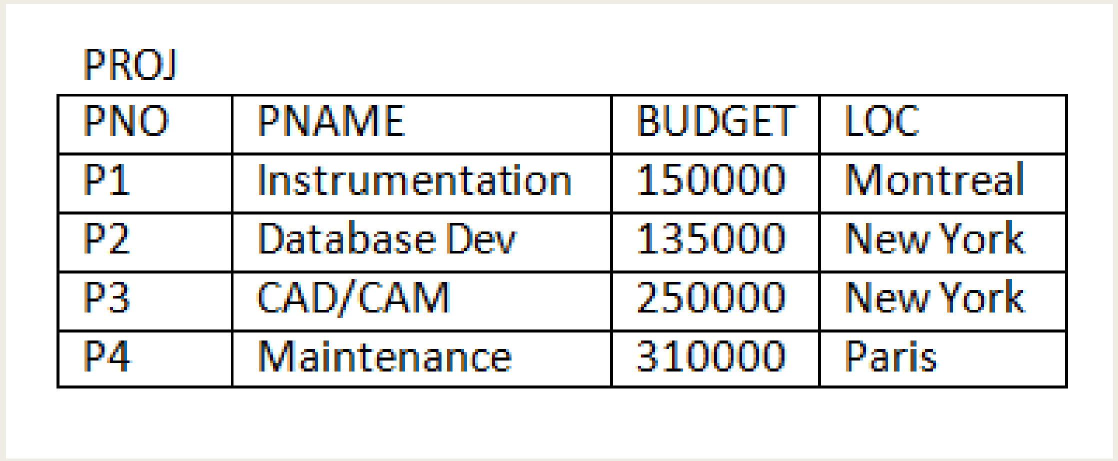
$Proj_1$

$\cup$

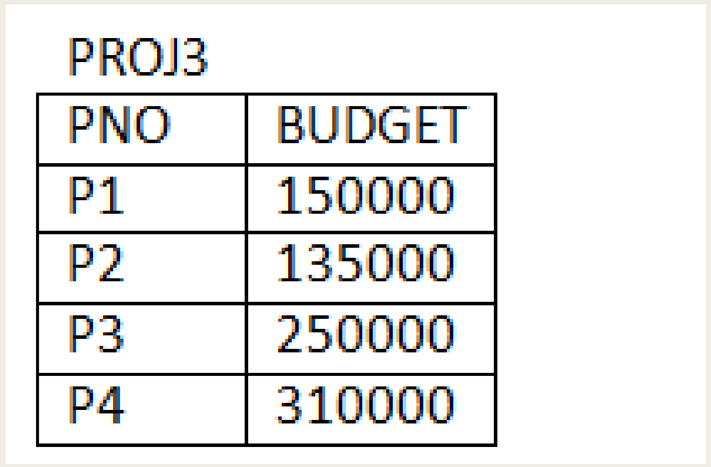
$Proj_2$

Question 2

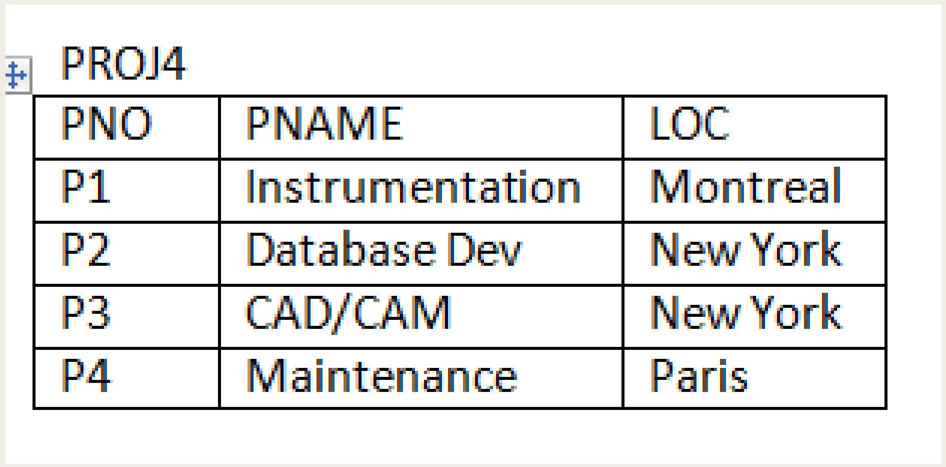
- Do a vertical fragmentation based on:
  - *PROJ3: information about project budgets.*
  - *PROJ4: information about project names and its locations*


PROJ

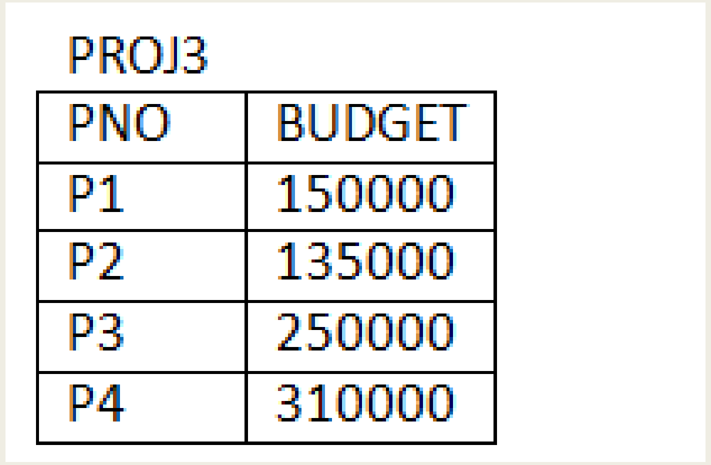
PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Dev	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris


PROJ3

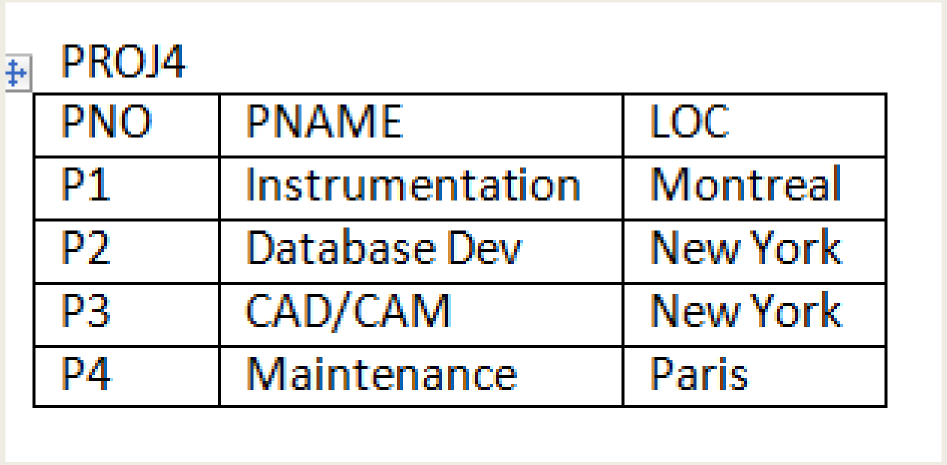
PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000


PROJ4

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Dev	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris


PROJ3

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000


PROJ4

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Dev	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris

For RA:

$PROJ_3 = \prod_{PNO,\ BUDGET} (PROJ)$

$PROJ_4 = \prod_{PNO,\ NAME,\ LOC} (PROJ)$

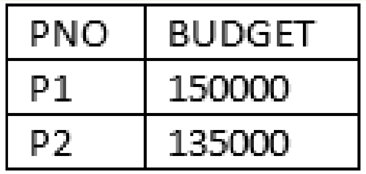
Reconstruction:

$PROJ_3 \bowtie_{PNO} PROJ_4$

Question 3

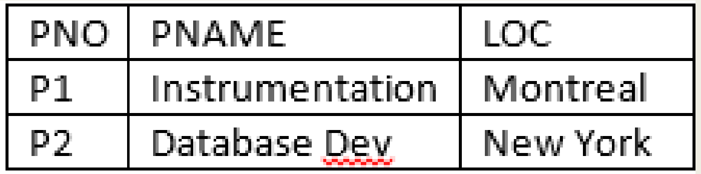
- Do a mixed fragmentation based on:
  - *PROJ1&3: information about project budgets and it must be less than $200,000*
  - *PROJ1&4: information about project names and its locations and it must be less than $200,000*
  - *PROJ2&3: information about project budgets and it must be greater than or equal $200,000*
  - *PROJ2&4: information about project names and its locations and it must be greater than or equal $200,000*

PROJ1&3



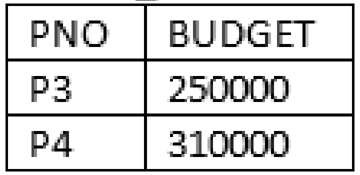
PNO	BUDGET
P1	150000
P2	135000

PROJ1&4



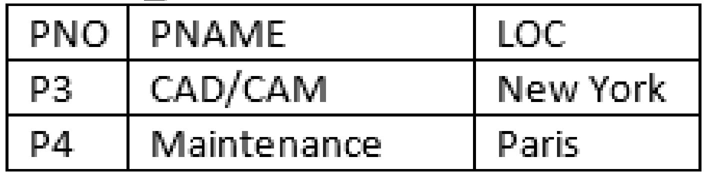
PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Dev	New York

PROJ2&3



PNO	BUDGET
P3	250000
P4	310000

PROJ2&4



PNO	PNAME	LOC
P3	CAD/CAM	New York
P4	Maintenance	Paris

For RA:

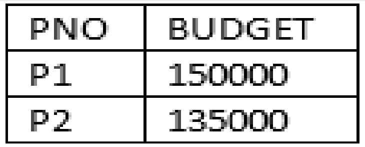
$$PROJ_{1\&3} = \Pi_{PNO,\ BUDGET}\ \sigma_{BUDGET<200K}\ (PROJ)$$

$$PROJ_{1\&4} = \Pi_{PNO,\ NAME,\ LOC}\ \sigma_{BUDGET<200K}\ (PROJ)$$

$$PROJ_{2\&3} = \Pi_{PNO,\ BUDGET}\ \sigma_{BUDGET>=200K}\ (PROJ)$$

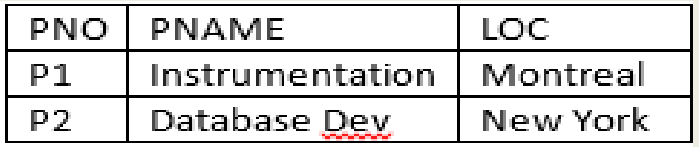
$$PROJ_{2\&4} = \Pi_{PNO,\ NAME,\ LOC}\ \sigma_{BUDGET>=200K}\ (PROJ)$$

PROJ1&3



PNO	BUDGET
P1	150000
P2	135000

PROJ1&4



PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Dev	New York

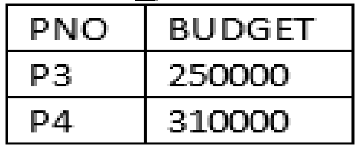
Reconstruction:

$$PROJ_{1\&3} \bowtie PROJ_{1\&4}$$

$$\cup$$

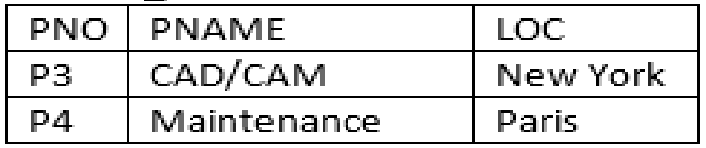
$$PROJ_{2\&3} \bowtie PROJ_{2\&4}$$

PROJ2&3



PNO	BUDGET
P3	250000
P4	310000

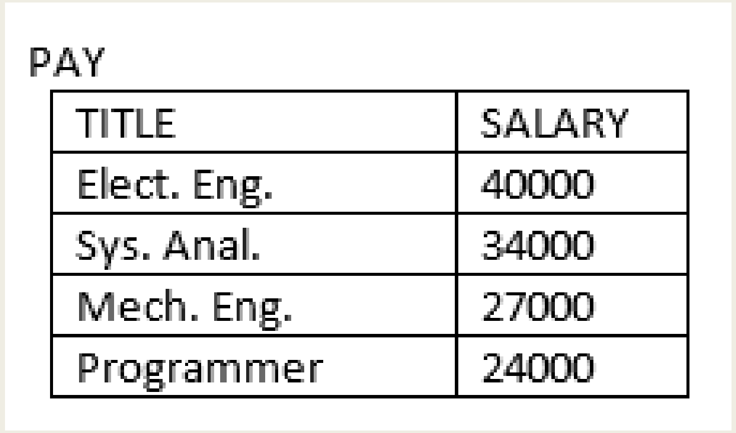
PROJ2&4



PNO	PNAME	LOC
P3	CAD/CAM	New York
P4	Maintenance	Paris

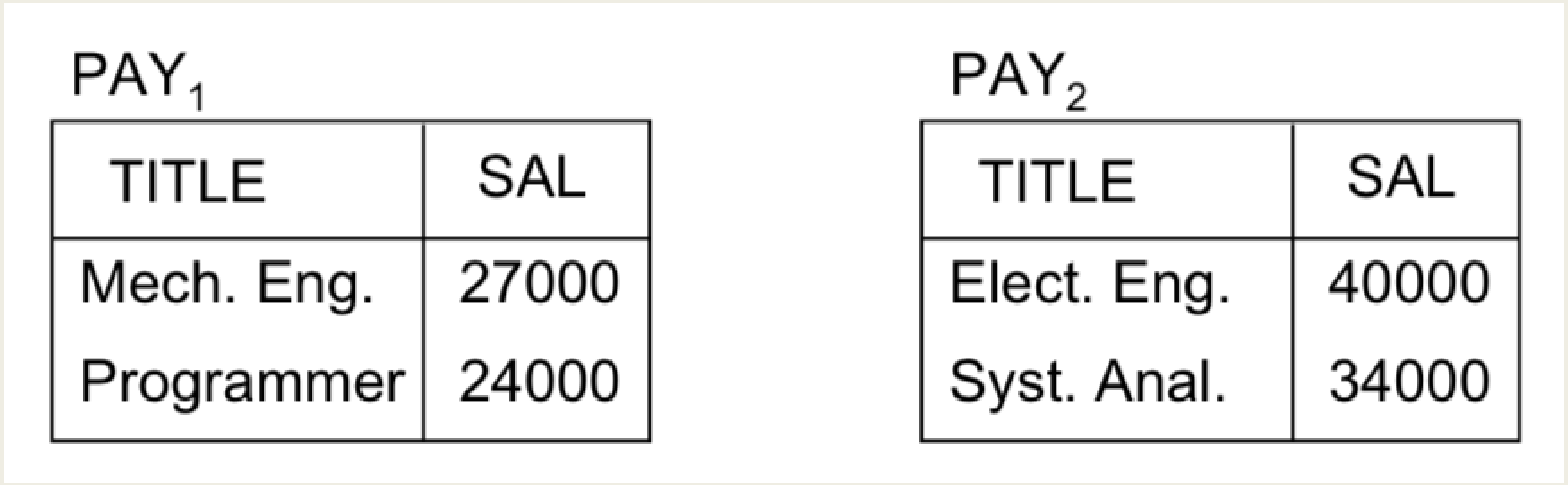
QUESTION 4

- Do a horizontal fragmentation based on:
  - PAY1: salary less than $30,000
  - PAY2: salary greater than $30,000



PAY

TITLE	SALARY
Elect. Eng.	40000
Sys. Anal.	34000
Mech. Eng.	27000
Programmer	24000

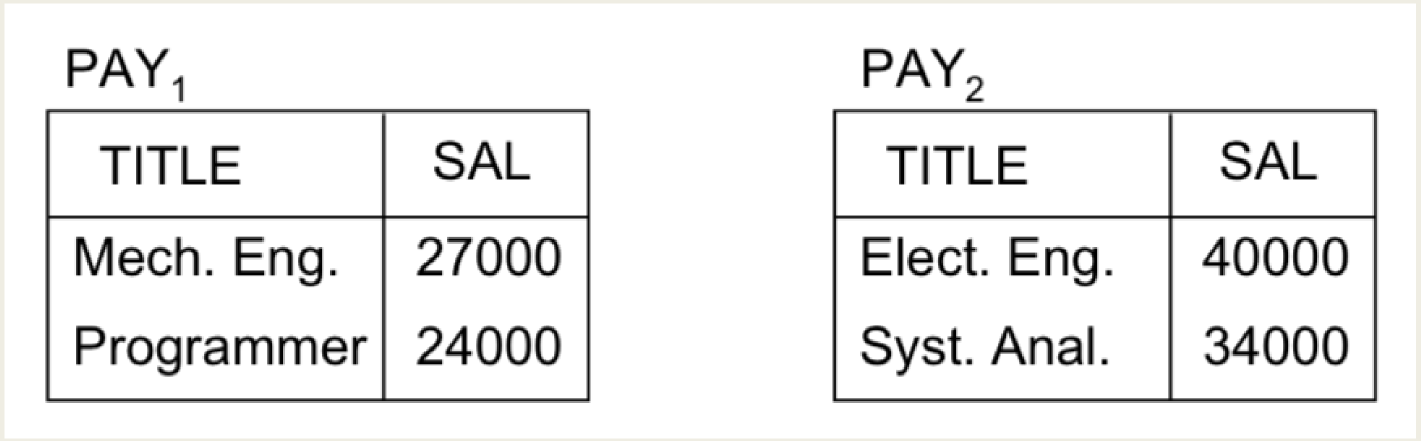


PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000



PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000

By using RA:

$Pay_1 = \sigma_{SALARY<30K} (PAY)$

$Pay_2 = \sigma_{SALARY>30K}(PAY)$

Reconstruction:

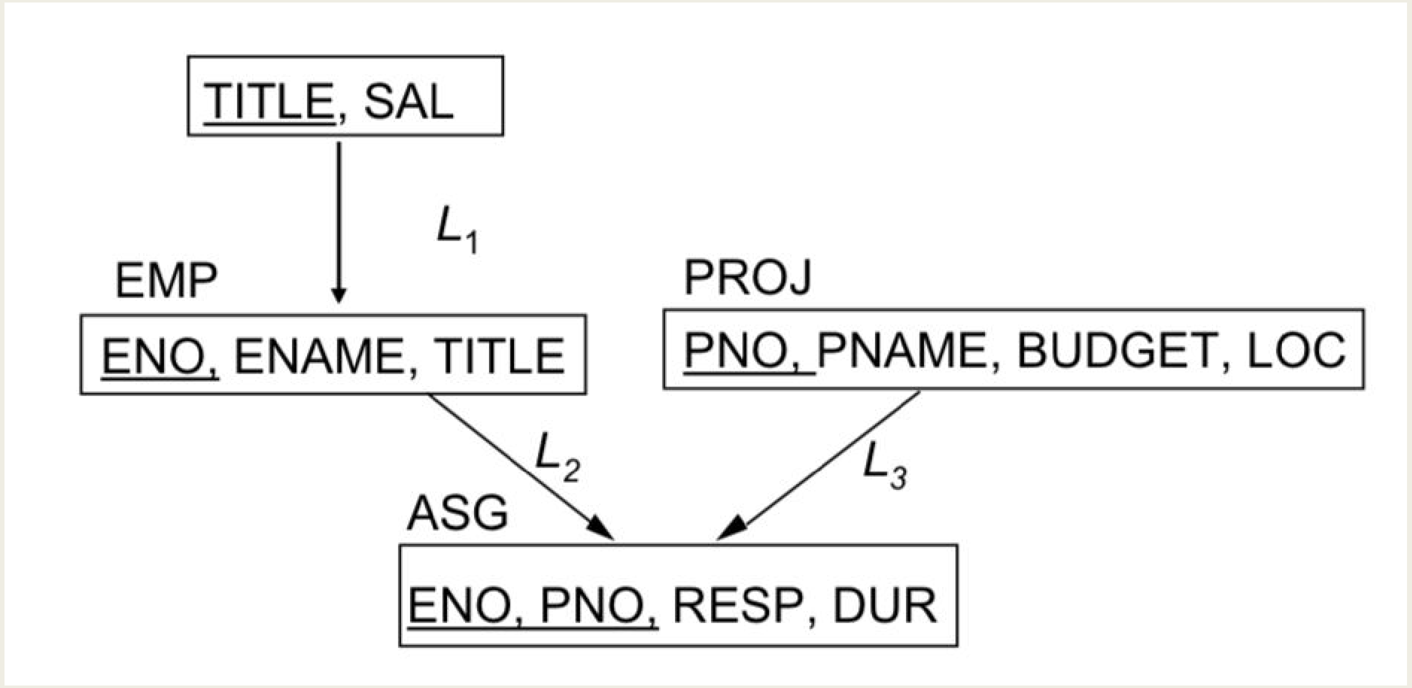
Pay_1

$\cup$

Pay_2

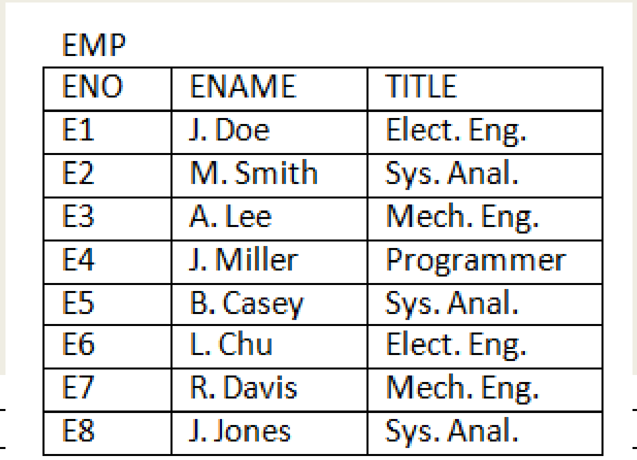
Question 5

- Identify which table is a CHILD table to PAY table.
  - EMPLOYEE



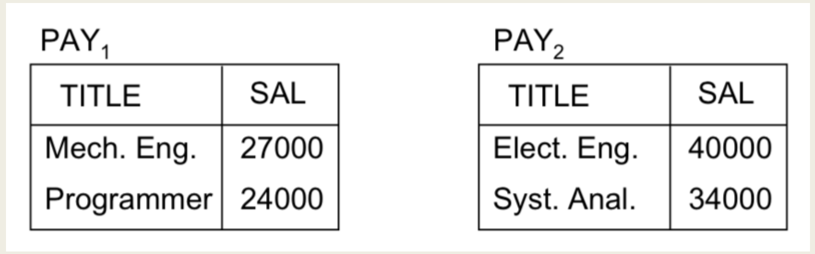
Question 6

- Do a derived fragmentation of an EMPLOYEE table.
  - EMP1: employee with salary less than $30,000
  - EMP2: employee with salary greater than $30,000



EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Sys. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Sys. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Sys. Anal.

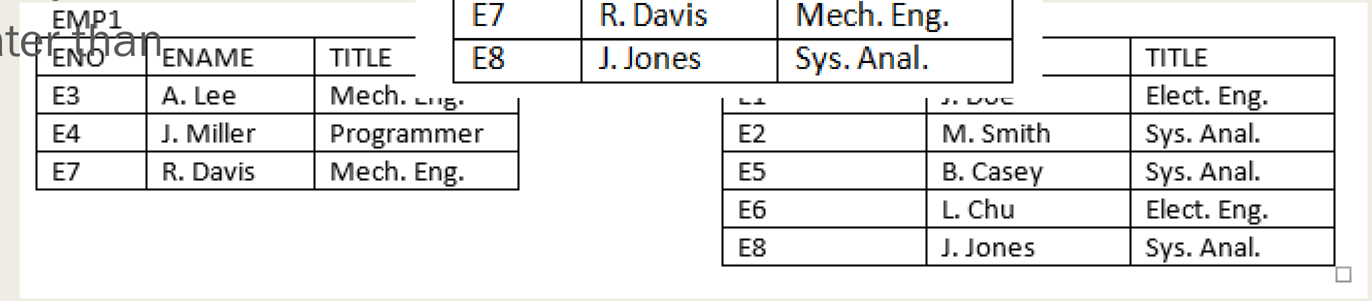


PAY$_1$

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY$_2$

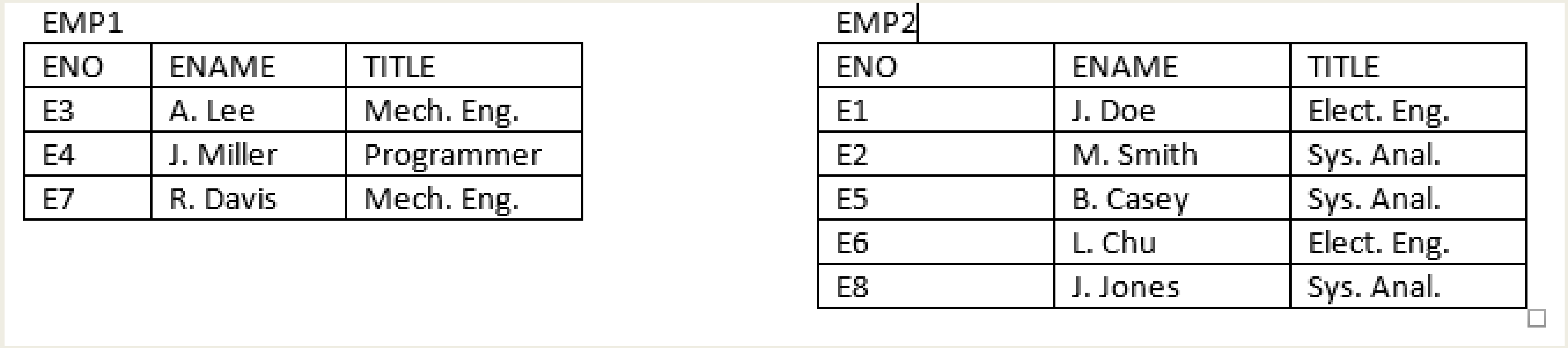
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000



EMP1

ENO	ENAME	TITLE
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E7	R. Davis	Mech. Eng.

		TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Sys. Anal.
E5	B. Casey	Sys. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Sys. Anal.



EMP1

ENO	ENAME	TITLE
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E7	R. Davis	Mech. Eng.

EMP2

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Sys. Anal.
E5	B. Casey	Sys. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Sys. Anal.

BY RA:

$$\mathrm{EMP}_1 = EMP \ltimes_{TITLE} PAY_1$$

$$\mathrm{EMP}_2 = EMP \ltimes_{TITLE} PAY_2$$

No Fragmentation

- A final strategy is not to fragment a relation.
- For example, the Branch relation contains
  only a small number of tuples and is not updated very frequently.
- Hence, it is better to leave the table that way as fragmenting it will lead to nothing better.

Correctness of Fragmentation

Completeness

If relation R is decomposed into fragments R_1, R_2, ... R_n, each data item that can be found in R must appear in at least one fragment.

Reconstruction

- Must be possible to define a relational operation that will reconstruct *R* from the fragments.
- Reconstruction for horizontal fragmentation is Union operation and Join for vertical .

Correctness of Fragmentation

Disjointness

- If data item d_i appears in fragment R_i, then it should not appear in any other fragment.
- Exception: vertical fragmentation, where primary key attributes must be repeated to allow reconstruction.
- For horizontal fragmentation, data item is a tuple.
- For vertical fragmentation, data item is an attribute.

Transparencies in a DDBMS

- Distribution Transparency
  - *Fragmentation Transparency*
  - *Location Transparency*
  - *Replication Transparency*
  - *Local Mapping Transparency*
  - *Naming Transparency*

- Transaction Transparency
- Performance Transparency
- DBMS Transparency

Distribution Transparency

- Allows management of a physically dispersed database as though it were a centralized database
- Supported by a distributed data dictionary (DDD) which contains the description of the entire database as seen by the DBA
  - *The DDD is itself distributed and replicated at the network nodes*
- Three levels of distribution transparency are recognized:
  - *Fragmentation transparency – user does not need to know if a database is partitioned; fragment names and/or fragment locations are not needed*
  - *Location transparency – fragment name, but not location, is required*
  - *Local mapping transparency – user must specify fragment name and location*

A Summary of Transparency Features

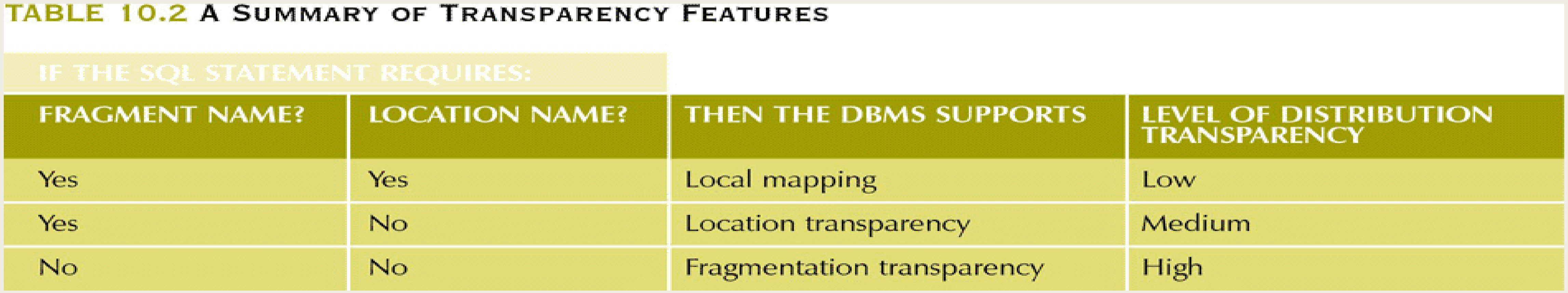
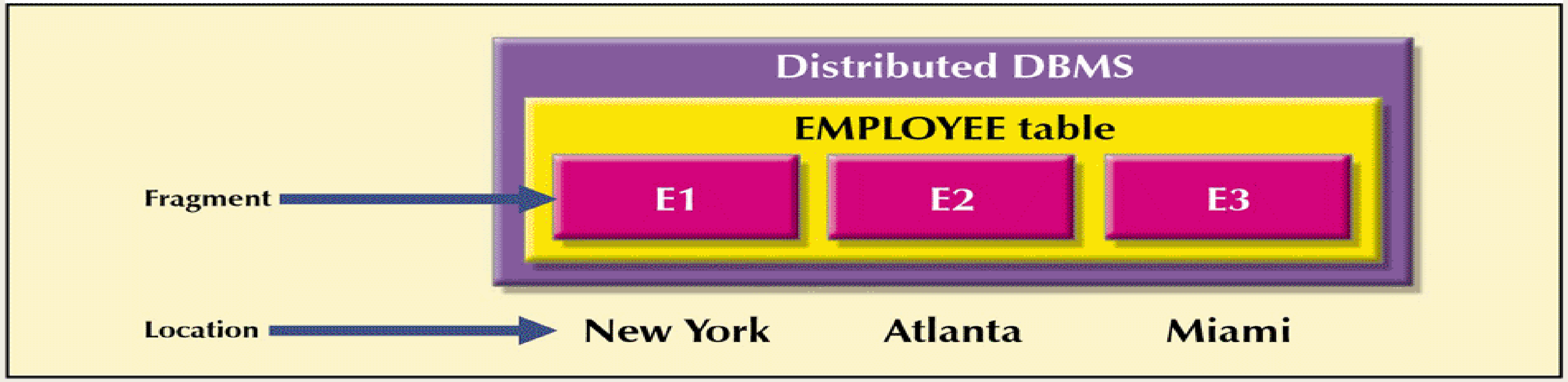


TABLE 10.2 A Summary of Transparency Features

IF THE SQL STATEMENT REQUIRES:			
FRAGMENT NAME?	LOCATION NAME?	THEN THE DBMS SUPPORTS	LEVEL OF DISTRIBUTION TRANSPARENCY
Yes	Yes	Local mapping	Low
Yes	No	Location transparency	Medium
No	No	Fragmentation transparency	High

FIGURE 10.9 Fragment Locations



Distribution Transparency

- The EMPLOYEE table is divided among three locations (no replication)

- Suppose an employee wants to find all employees with a birthdate prior to jan 1, 1940

  - *Fragmentation transparency-*
    - SELECT * FROM EMPLOYEE WHERE EMP_DOB < '01-JAN-1940';

  - *Location transparency-*
    - SELECT * FROM E1 WHERE EMP_DOB < '01-JAN-1940' UNION SELECT * FROM E2 … UNION SELECT * FROM E3…;

  - *Local Mapping Transparency*
    - SELECT * FROM E1 NODE NY WHERE EMP_DOB < '01-JAN-1940' UNION SELECT * FROM E2 NODE ATL … UNION SELECT * FROM E3 NODE MIA…;

Naming Transparency

- Each item in a DDB must have a unique name.
- DDBMS must ensure that no two sites create a database object with same name.
- One solution is to create central name server. However, this results in:
  - *loss of some local autonomy;*
  - *central site may become a bottleneck;*
  - *low availability; if the central site fails, remaining sites cannot create any new objects.*

Replication Transparency

- Replication Transparency
  - *With replication transparency, user is unaware of replication of fragments .*

Transaction Transparency

- Ensures database transactions will maintain distributed database's integrity and consistency

- A DDBMS transaction can update data stored in many different computers connected in a network
  - *Transaction transparency ensures that the transaction will be completed only if all database sites involved in the transaction complete their part of the transaction*

Performance transparency

- *Performance transparency – allows system to perform as if it were a centralized DBMS.*

- *No performance degradation due to use of a network or platform differences*

DBMS Transparency

- DBMS transparency hides the knowledge that the local DBMSs may be different, and is therefore only applicable to heterogeneous DDBMSs.

- It is one of the most difficult transparencies to provide as a generalization.

References

Database Systems: A Practical Approach to Design, Implementation, and Management, Thomas Connolly and Carolyn Begg, 5th Edition, 2010, Pearson.