INT206

Distributed DBMS

Sanit Sirisawatvatana and Sunisa Sathapornwajana

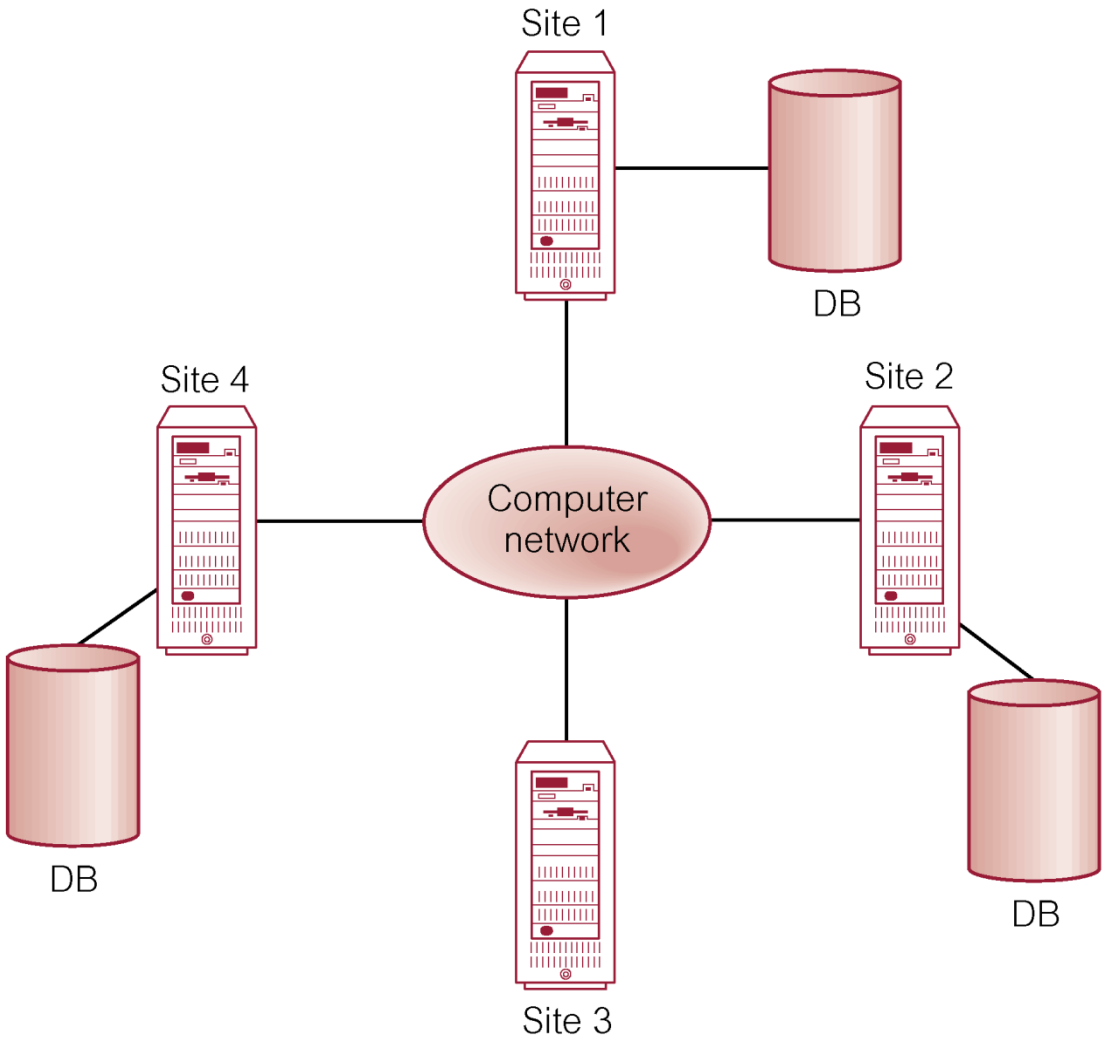
Objectives

- Concepts.
- Advantages and disadvantages of distributed databases.
- Functions and architecture for a DDBMS.
- Distributed database design.
- Levels of transparency.
- Comparison criteria for DDBMSs.

Concepts

- Distributed Database
  - A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network.
- Distributed DBMS
  - Software system that permits the management of the distributed database and makes the distribution transparent to users.
- The objective of transparency is to make the distributed system appear like a centralized system

Distributed DBMS



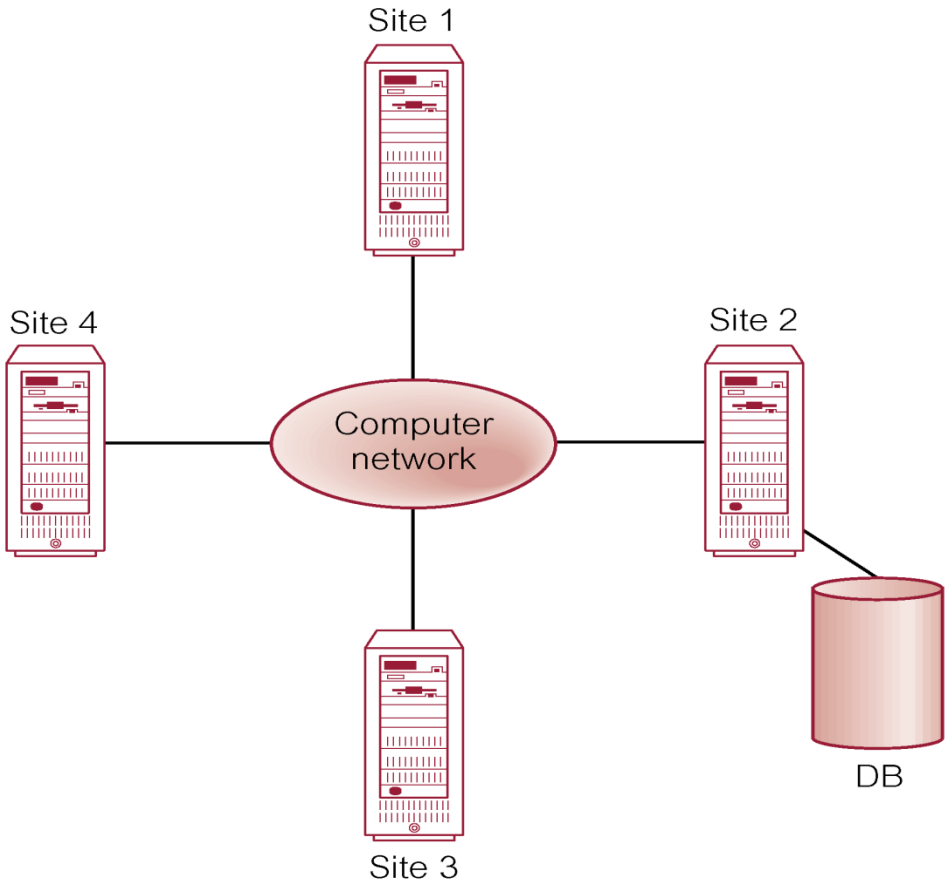
DDBMS Characteristics

- A collection of logically related shared data
- The data is split into a number of fragments
- Fragments may be replicated
- Fragments/replicas are allocated to sites
- The sites are linked by a communications network
- The data at each site is under the control of a DBMS
- The DBMS at each site can handle local applications autonomously
- Each DBMS participates in at least one global application.

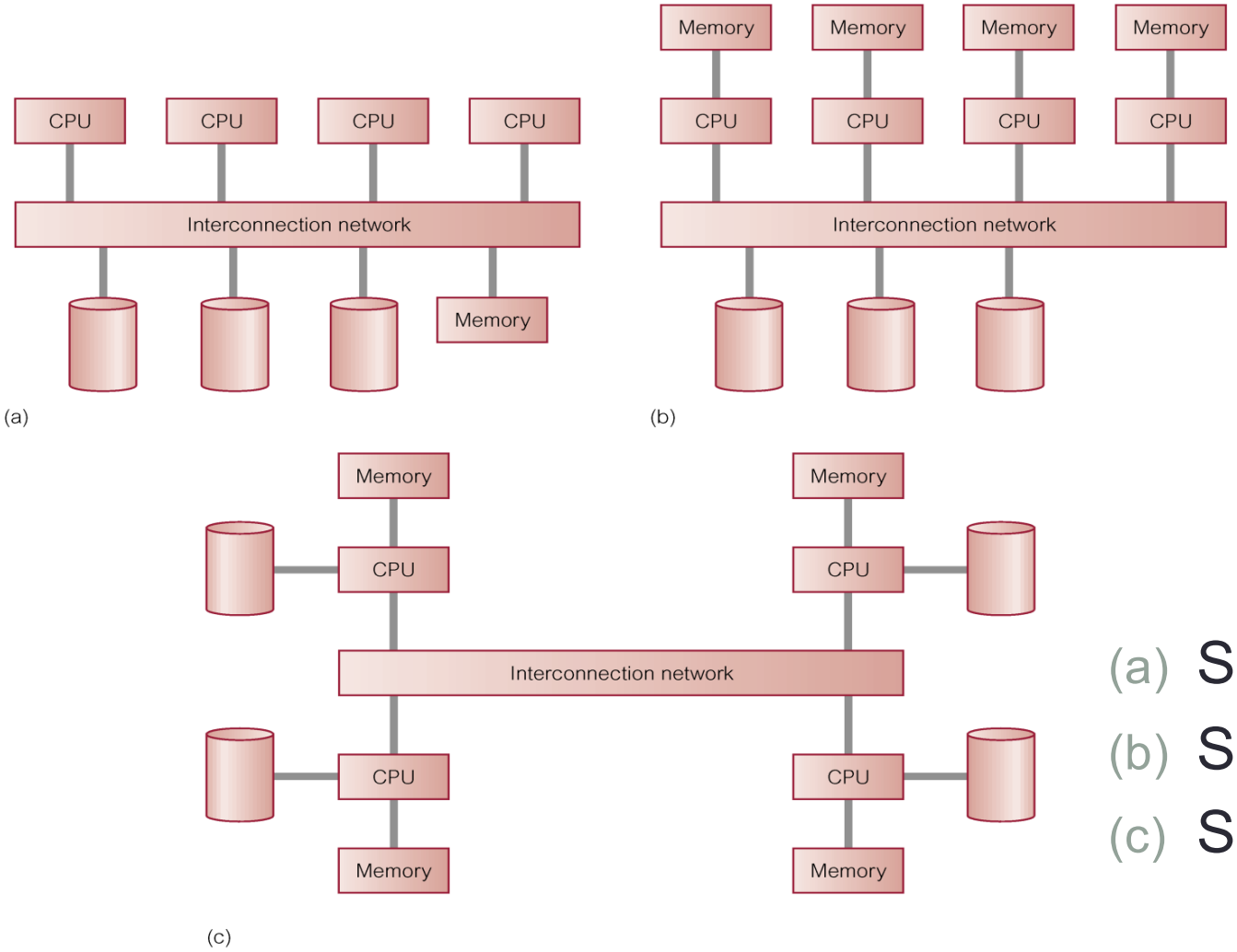
Concepts

- Distributed processing
  - A centralized database that can be accessed over a computer network
- Parallel DBMSs
  - A DBMS running across multiple processors and disks that is designed to execute operations in parallel in order to improve performance
  - Parallel DBMSs link multiple, smaller machines to achieve the same throughput as single, larger machine often with greater scalability and reliability than single-processor DBMSs.
  - Parallel DBMSs provide shared resource management for a single database

Distributed Processing



Parallel database architecture



(a) Shared memory

(b) Shared disk

(c) Shared nothing

Advantages of DDBMSs

- Reflects organizational structure
- Improved shareability and local autonomy
- Improved availability
- Improved reliability
- Improved performance
- Economics
- Modular growth

Disadvantages of DDBMSs

- Complexity
- Cost
- Security
- Integrity control more difficult
- Lack of standards
- Lack of experience
- Database design more complex

Types of DDBMSs

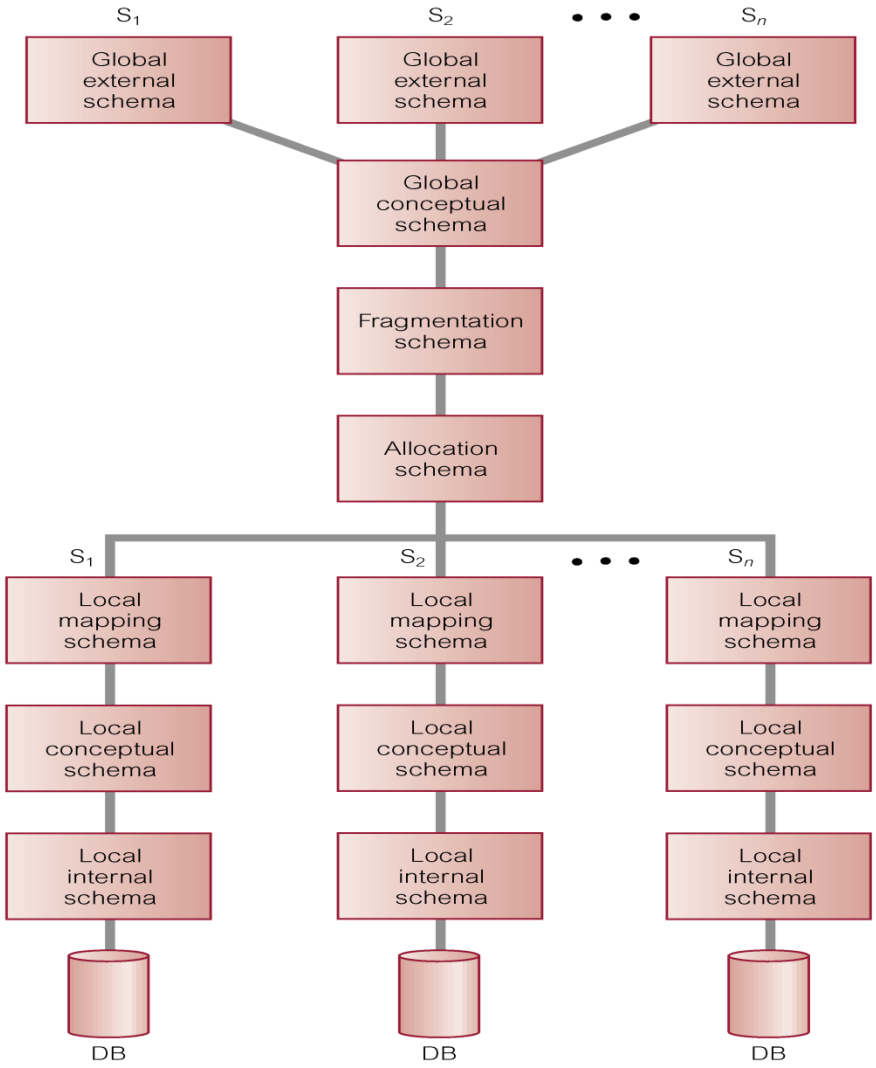
- Homogeneous DDBMSs
  - All sites use same DBMS product.
  - Much easier to design and manage.
  - Approach provides incremental growth and allows increased performance.
- Heterogeneous DDBMSs
  - Sites may run different DBMS products, with possibly different underlying data models.
  - Occurs when sites have implemented their own databases and integration is considered later.
  - Typical solution is to use gateways, which convert the language and model of each different DBMS into the language and model of the relational system.

Functions of DDBMSs

DDBMSs are expected to have following functionality:

- Extended communication services.
  - To transfer query and data among the sites using a network
- Extended Data Dictionary
  - To store data distribution detail
- Distributed query processing
  - Including query optimization and remote data access
- Extended security control
  - To maintain appropriate authorization/access privileges
- Extended concurrency control
  - To maintain consistency of distributed/ replicated data
- Extended recovery services
  - To recover failures of individual sites and communication links

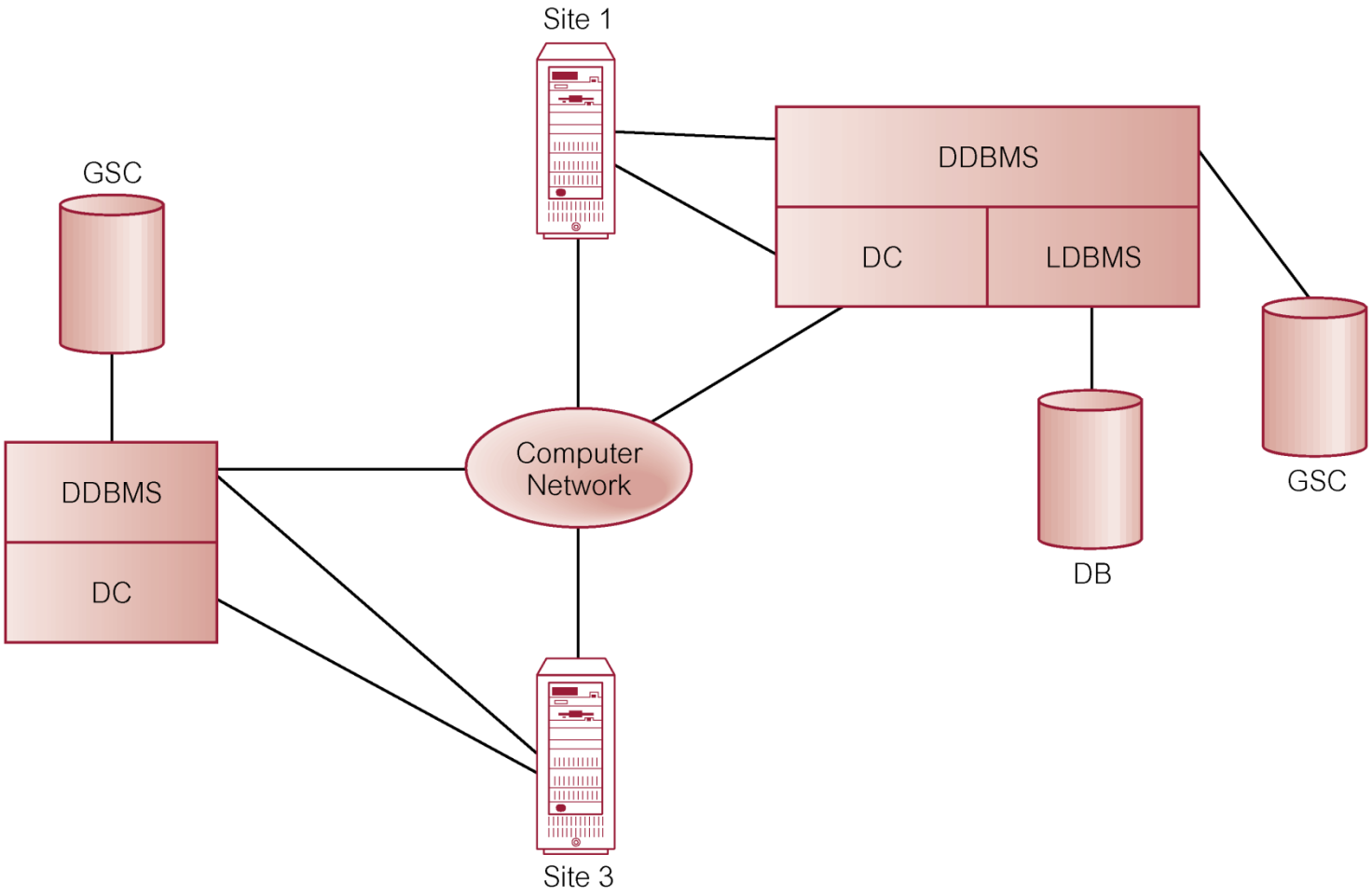
Architecture of a DDBMS



Architecture of a DDBMS

- Global conceptual schema
  - Is a logical description of the whole database
  - Contains definitions of entities, relationships, constraints, security and integrity information
- Fragmentation schema
  - Is a description of how the data is to be logically partitioned
- Allocation schema
  - Is a description of where the data is to be located taking account of any replication
- Local schema
  - Each DBMS has its own set of schema
- Local mapping schema
  - Maps fragments in the allocation schema into external objects in the local database

Component Architecture for a DDBMS



Component Architecture for a DDBMS

Four major components:

- Local DBMS (LDBMS) component
  - Is a standard DBMS to control the local data at each site that has a database
  - Has its own local system catalog
- Data communications (DC) component
  - Is the software that enables all sites to communicate with each other
  - Contains information about the sites and the link
- Global system catalog (GSC)
  - Is the system catalog of distributed system
  - Holds information such as the fragmentation, replication and allocation schema
- Distributed DBMS (DBMS) component
  - Is the controlling unit of entire system

Distributed Relational Database Design

Three main concepts

- Fragmentation
  - A relation may be divided into a number of subrelations, called fragments, which are then distributed
  - Can be horizontal or vertical fragments
- Allocation
  - Each fragment is stored at the site with optimal distribution
- Replication
  - DDBMS may maintain a copy of a fragment at several different sites

Distributed Relational Database Design

The definition and allocation of fragments must focus to achieve the following objectives:

- Locality of reference
  - Store data close to where it is used
- Improved reliability and availability
  - Is improved by replication
- Acceptable performance
  - Avoid the bottlenecks and underutilization of resources
- Balanced storage capacities and costs
- Minimal communication costs

Data allocation

Four strategies for the placement of data

- Centralized
  - Locality of reference is at the lowest as all sites (except the central site)
  - Communication cost is high
- Fragmented (or partitioned)
  - Locality of reference is high
  - Storage costs are low (no replication)
  - Reliability and availability are low
  - Performance should be good and communication costs low

Data allocation

- Complete replication
  - Locality of reference, reliability and availability and performance are maximized
  - Storage costs and communication cost are the most expensive
- Selective replication
  - Is a combination of fragmentation, replication, and centralization
  - Some data item are fragmented to achieve high locality of reference
  - Data are used at many sites and are not frequently updated are replicated
  - Other data items are centralized
  - Is the most common used strategy because of its flexibility

Comparison of strategies for data allocation

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.

Fragmentation

Reasons for fragmenting a relation

- Usage
  - Application uses some data rather than entire relations
- Efficiency
  - Data is stored close to where it is most frequently used
- Parallelism
  - Allow a transaction can be divided into several subqueries that operate on fragments
- Security
  - Data not required by local application is not stored and not available to unauthorized users

Fragmentation

It has two primary disadvantages

- Performance
  - Global applications require data from several fragments located at different sites may be slower
- Integrity
  - Integrity control may be more difficult because of fragmented data at different sites

Correctness of fragmentation

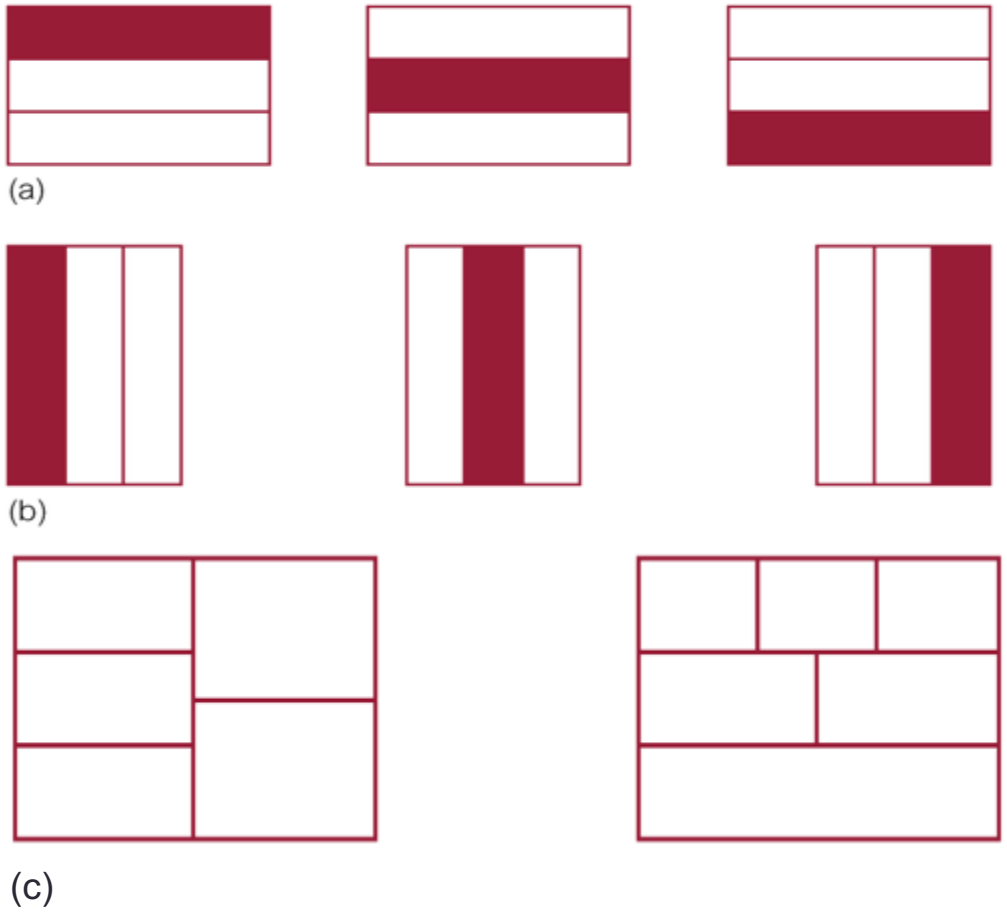
Three rules that must be followed during fragmentation:

- Completeness (no loss of data)
  - $R = R_1 \cup R_2 \cup R_3 \ \ldots \cup R_n$
  - Each R_i fragment must be appear at least one fragment
- Reconstruction (functional dependencies)
  - Must be possible to define a relational operation to reconstruct R from the fragments
  - Horizontal fragment –> Union operation
  - Vertical fragment -> Join operation
- Disjointness (minimal data redundancy)
  - $R_i \cap R_j = \{\}$ empty set (i = 1..n, j = 1..n , i<>j)
  - No data item in fragment appear in any other fragment
  - Except for the vertical fragmentation, the primary key attributes are repeated for reconstruction

Types of Fragmentation

Four types of fragmentation

a) Horizontal
b) Vertical
c) Mixed
d) Derived



Horizontal Fragmentation

- Consists of a subset of the tuples of a relation.
- Defined using *Selection* operation of relational algebra:

$$\sigma_p(R)$$

- For example:

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

Completeness : $R = P_1 \cup P_2$

Reconstruction: Using the Union operation

Disjointness: No property type that is both 'House' and 'Flat'

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.

- Completeness : $R = S_1 \bowtie S_2$
- Reconstruction: Using the Join operation
- Disjointness: Fragments are disjoint except for the primary key for reconstruction

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:

$$\sigma_{p}(\Pi_{a1, \ldots, an}(R)) \quad \text{or}$$
$$\Pi_{a1, \ldots, an}(\sigma_{p}(R))$$

Example - Mixed Fragmentation

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

$S_{21} = \sigma_{branchNo='B003'}(S_2)$
$S_{22} = \sigma_{branchNo='B005'}(S_2)$
$S_{23} = \sigma_{branchNo='B007'}(S_2)$

- Completeness : $R = S_1 \bowtie (S_{21} \cup S_{22} \cup S_{23})$
- Reconstruction: Using the Union and Join operation
- Disjointness: Fragments are disjoint; No staff member who works in more than one branch and S_1 and S_2 are disjoint except for the duplication of primary key

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:

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

Example - Derived Horizontal Fragmentation

$S_3 = \sigma_{branchNo='B003'}(Staff)$

$S_4 = \sigma_{branchNo='B005'}(Staff)$

$S_5 = \sigma_{branchNo='B007'}(Staff)$

Could use derived fragmentation for Property:

$P_i = PropertyForRent \rhd_{branchNo} S_i, \quad 3 \leq i \leq 5$

Distributed Database Design Methodology

1. Use normal methodology to produce a design for the global relations.
2. Examine topology of system to determine where databases will be located.
3. Analyze most important transactions and identify appropriateness of horizontal/vertical fragmentation.
4. Decide which relations are not to be fragmented.
5. Examine relations on 1 side of relationships and determine a suitable fragmentation schema. Relations on many side may be suitable for derived fragmentation.

Transparencies in DDBMS

- Definition of DDBMS
  - The system should make distribution transparent to the user
- Transparency hides implementation details form the user
- DDBMS may provide various levels of transparency.
  - To make use of the distributed database equivalent to that of a centralized database
- Four main types of transparency in a DDBMS
  - Distribution transparency
  - Transaction transparency
  - Performance transparency
  - DBMS transparency

Distribution Transparency

- Allows the user to perceive the database as single, logical entity

- Fragmentation transparency
  - The user does not need to know the data is fragmented
- Location transparency
  - The user needs to know that the data is fragmented but does not need to know the location of data items
- Replication transparency
  - The user is unaware of the replication of fragments
  - Is closely related to location transparency
- Local mapping transparency
  - The user needs to know that the data is fragmented and the location of fragments

Example

- Fragmentation transparency

```
SELECT fName, lName
FROM Staff
WHERE position = 'Manager';
```

- Location transparency

```
SELECT fName, lName
FROM S21
WHERE staffNo IN (SELECT staffNo FROM S1 WHERE position = 'Manager')
UNION
SELECT fName, lName
FROM S22
WHERE staffNo IN (SELECT staffNo FROM S1 WHERE position = 'Manager')
UNION
SELECT fName, lName
FROM S23
WHERE staffNo IN (SELECT staffNo FROM S1 WHERE position = 'Manager');
```

Example

- Local mapping transparency

```
SELECT fName, lName
      FROM S21@SITE3
      WHERE staffNo IN (SELECT staffNo FROM S1@SITE5 WHERE
       position = 'Manager')
UNION
SELECT fName, lName
       FROM S22@SITE5
       WHERE staffNo IN (SELECT staffNo FROM S1@SITE5 WHERE
        position = 'Manager')
UNION
SELECT fName, lName
       FROM S23@SITE7
       WHERE staffNo IN (SELECT staffNo FROM S1@SITE5 WHERE
        position = 'Manager')
```