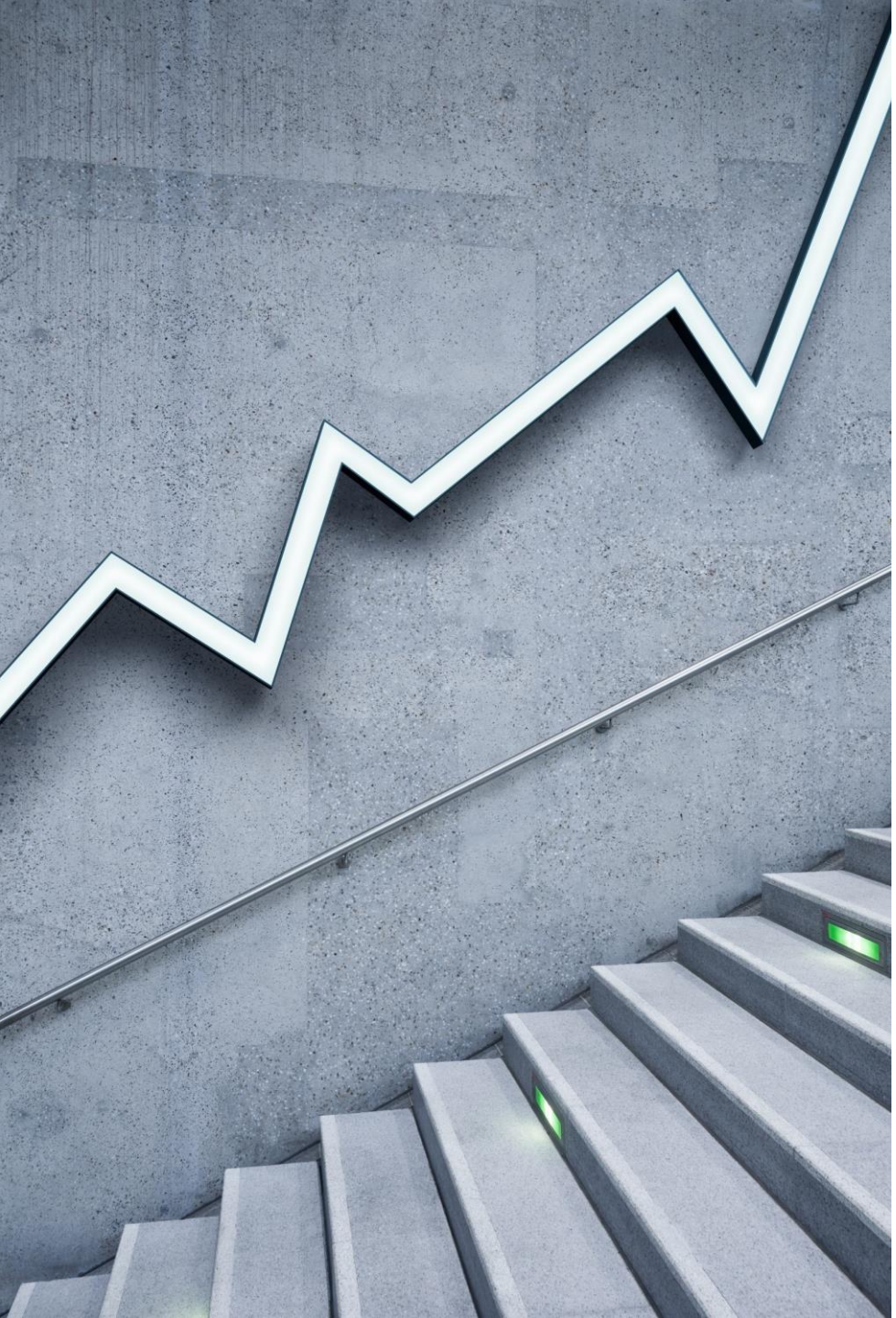


ملاحظة: تجدون فصلا كاملا خاص بهذا العرض في مطوية الدروس على منصة موودل

Distributed Databases

RABAH MOKHTARI

Introduction

Distributed database system (DDBS) technology is the union of two approaches to data processing: **database system** and **computer network technologies**.

1- Database systems

Database systems have taken us from a paradigm of data processing in which each application defined and maintained its own data to one in which the data are defined and administered centrally.

2- Computer network technologies

The technology of *computer networks*, on the other hand, promotes a mode of work that goes against all centralization efforts.

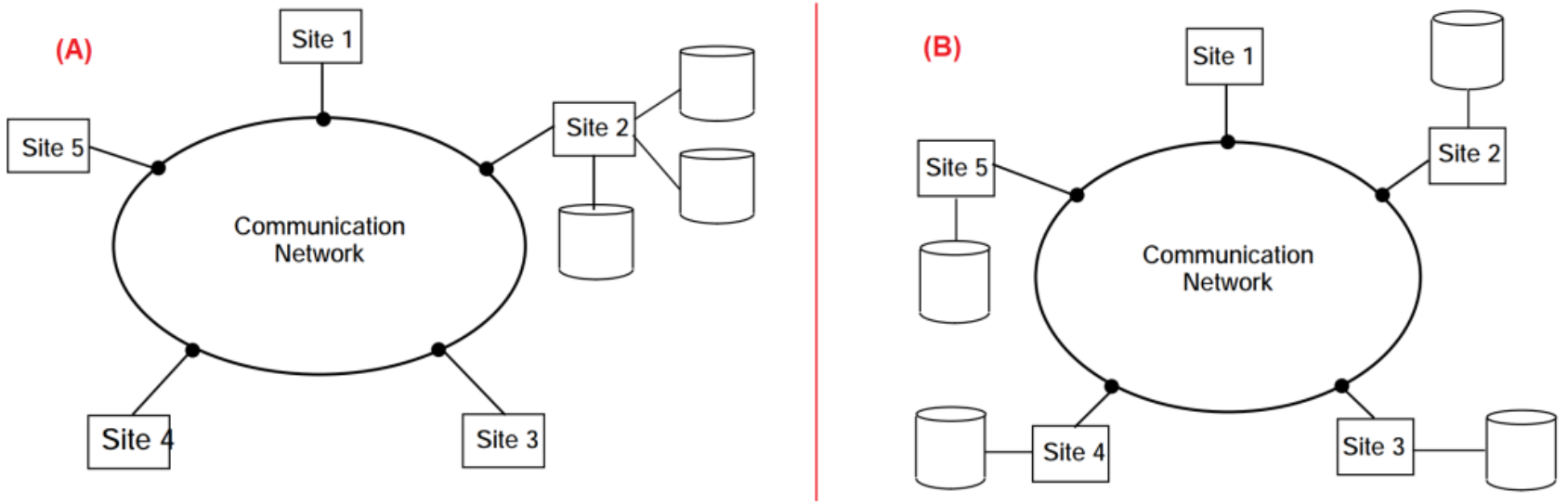
Distributed Data Processing

- Distributed data processing is a computing model in which data processing is distributed across multiple computers or nodes in a network.
- The processing can be done in parallel, allowing for faster and more efficient processing of large amounts of data.
- Each node in the network has access to a subset of the data, and the nodes work together to process the data and generate the desired output.

Distributed Database system

- A distributed database system is a type of database system that is spread across multiple computers geographically distributed.
- In a distributed database system, the data is **partitioned** or **replicated** across multiple nodes, and the nodes work together to process queries and transactions from clients.
- A DDBS is also not a system where, despite the existence of a network, the database resides at only one node of the network.

Distributed Database system



Centralized vs Distributed Database Systems: **(A)** Centralized Database System on network, **(B)** DDBS environment.

DDBS benefits

- **Scalability:** Distributed database systems can scale horizontally by adding more nodes to the network. This allows the system to handle large volumes of data and high transaction rates.
- **Fault tolerance:** Distributed database systems can continue to operate even if one or more nodes fail. Data can be replicated across multiple nodes, so if one node fails, another node can take over without loss of data.
- **Improved performance:** By distributing the data and processing across multiple nodes, distributed database systems can improve performance by processing queries and transactions in parallel.

Distributed DBMS architecture

- The architecture of a system defines its **structure**.
- This means that the components of the system are **identified**, the function of each component is **specified**, and the **interrelationships** and interactions among these components are defined.
- The specification of the architecture of a system requires identification of the various modules, with their interfaces and interrelationships, in terms of the data and control flow through the system.

ANSI/SPARC Architecture

- ANSI/SPARC Architecture is an early milestone in the field of database systems
- It was developed by the American National Standards Institute (ANSI) and the Standards Planning and Requirements Committee (SPARC) in the 1970s, when the field of database management was still in its early stages.
- It helped to establish many of the fundamental concepts and principles that are still used today.
- The ANSI/SPARC architecture defines three levels of abstraction for a database system

ANSI/SPARC Architecture

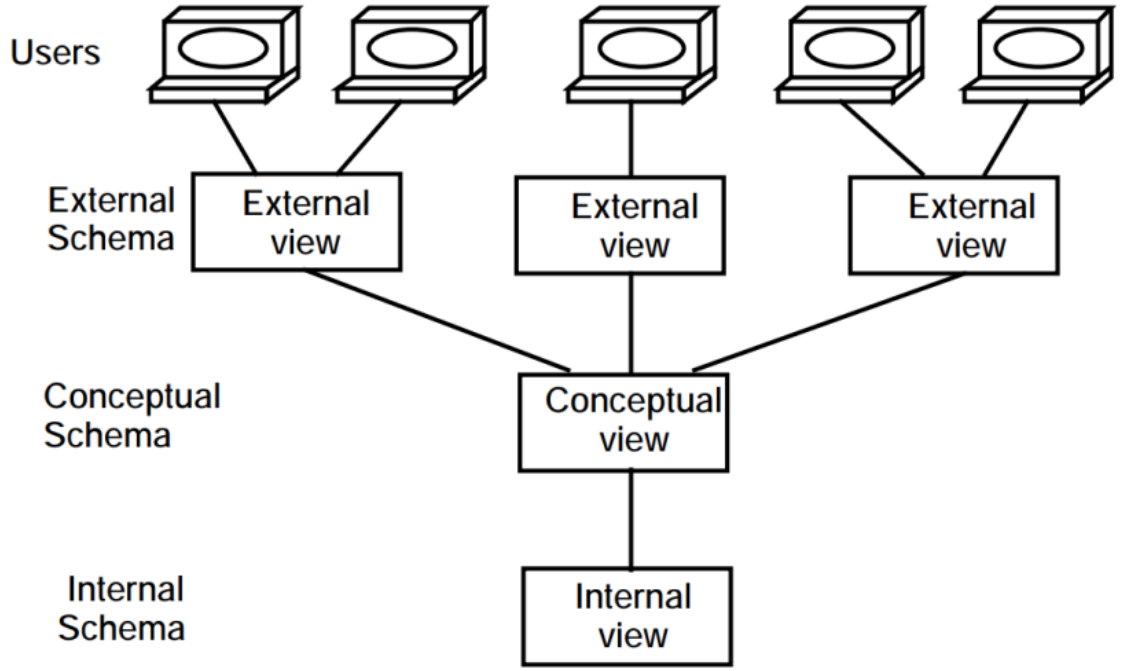


Figure 31: ANSI/SPARC Architecture

ANSI/SPARC Architecture

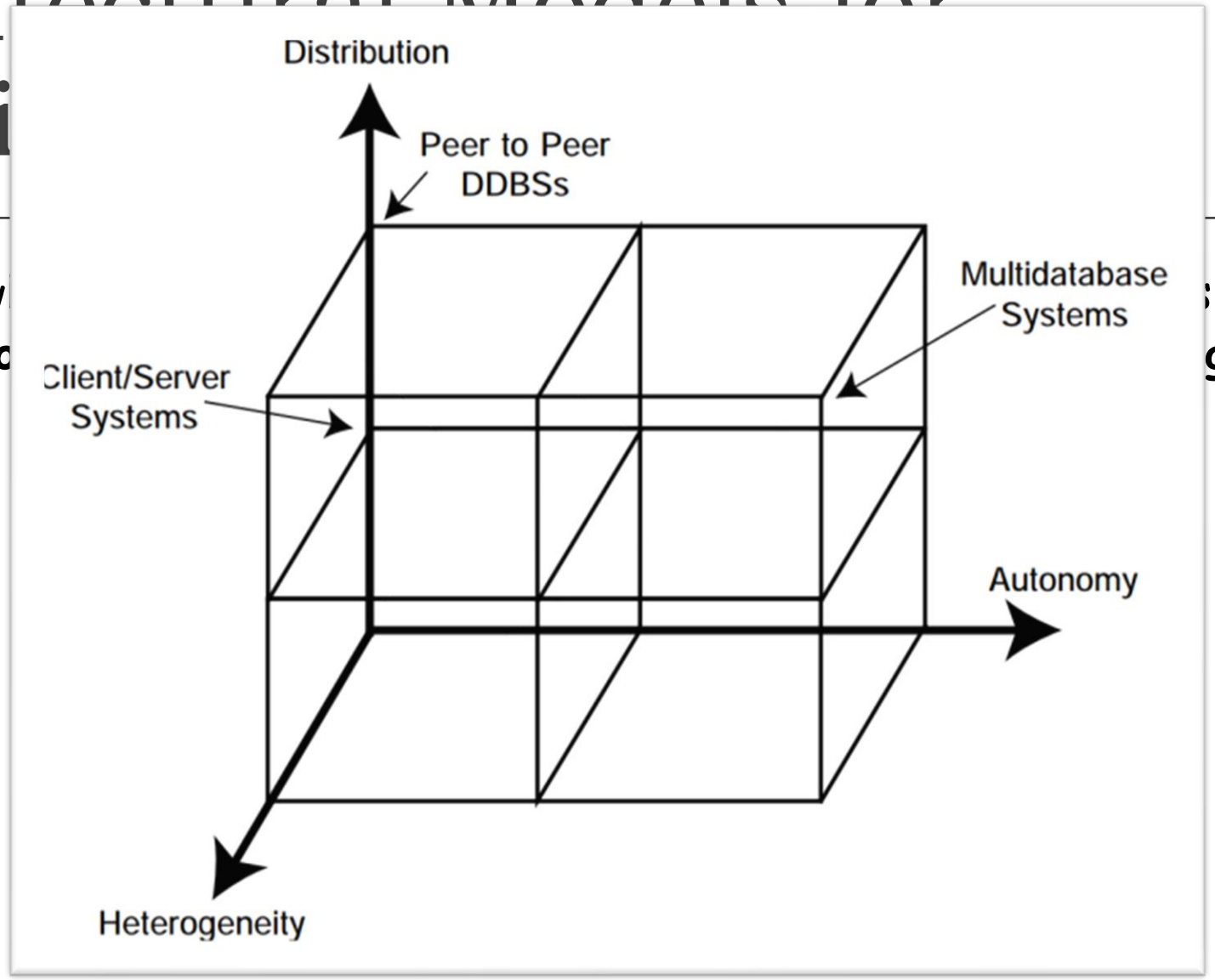
- **External level:** It describes how data is viewed by different users and groups, and how data is accessed and manipulated by applications. Each external schema is tailored to meet the specific needs of a particular user or application.

- **Conceptual level:** This is the level of the database system that describes the overall logical structure of the database. The conceptual schema is independent of any particular application or user, and is used to ensure that all data in the database is consistent and integrated

- **Internal level:** This is the level of the database system that describes how data is physically stored and accessed by the computer system. It defines the storage structures and access methods used by the DBMS to manage the data.

Architectural Models for Distri

The ways in w ified in terms of: the **autono geneity**.



Architectural Models for Distributed DBMSs

Autonomy

Autonomy refers to the distribution of **control**, not of data. It indicates the degree to which individual DBMSs can operate independently.

- The local operations of the individual DBMSs are not affected by their participation in the distributed system.
- The manner in which the individual DBMSs process queries and optimize them should not be affected by the execution of global queries that access multiple databases.
- System consistency or operation should not be compromised when individual DBMSs join or leave the distributed system.

Architectural Models for Distributed DBMSs

Distribution

- Distribution refers to the distribution of **data** over multiple sites.
- There are two alternatives classes: **client/server** distribution and **peer-to-peer** distribution (or **full distribution**).

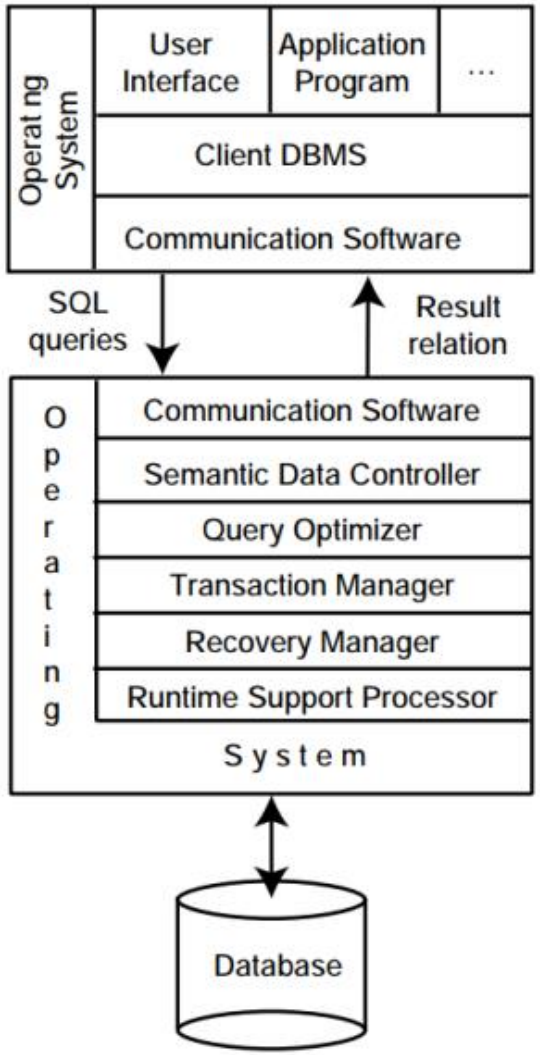
Heterogeneity

- Heterogeneity refers to the presence of diversity or differences in a distributed database environment in terms of data models, query languages, and transaction management protocols.

Client/Server architecture

- Client/server DBMSs entered the computing scene at the beginning of 1990s and have made a significant impact on both the DBMS technology and the way we do computing.
- the functions are divided into two classes: server functions and client functions.
- This provides a two-level architecture which makes it easier to manage the complexity of modern DBMSs and the complexity of distribution.
- We can cite many examples of DDBMS that use client/server architecture of distributed database systems. One such example is Microsoft SQL Server, Oracle Database, MySQL and PostgreSQL.

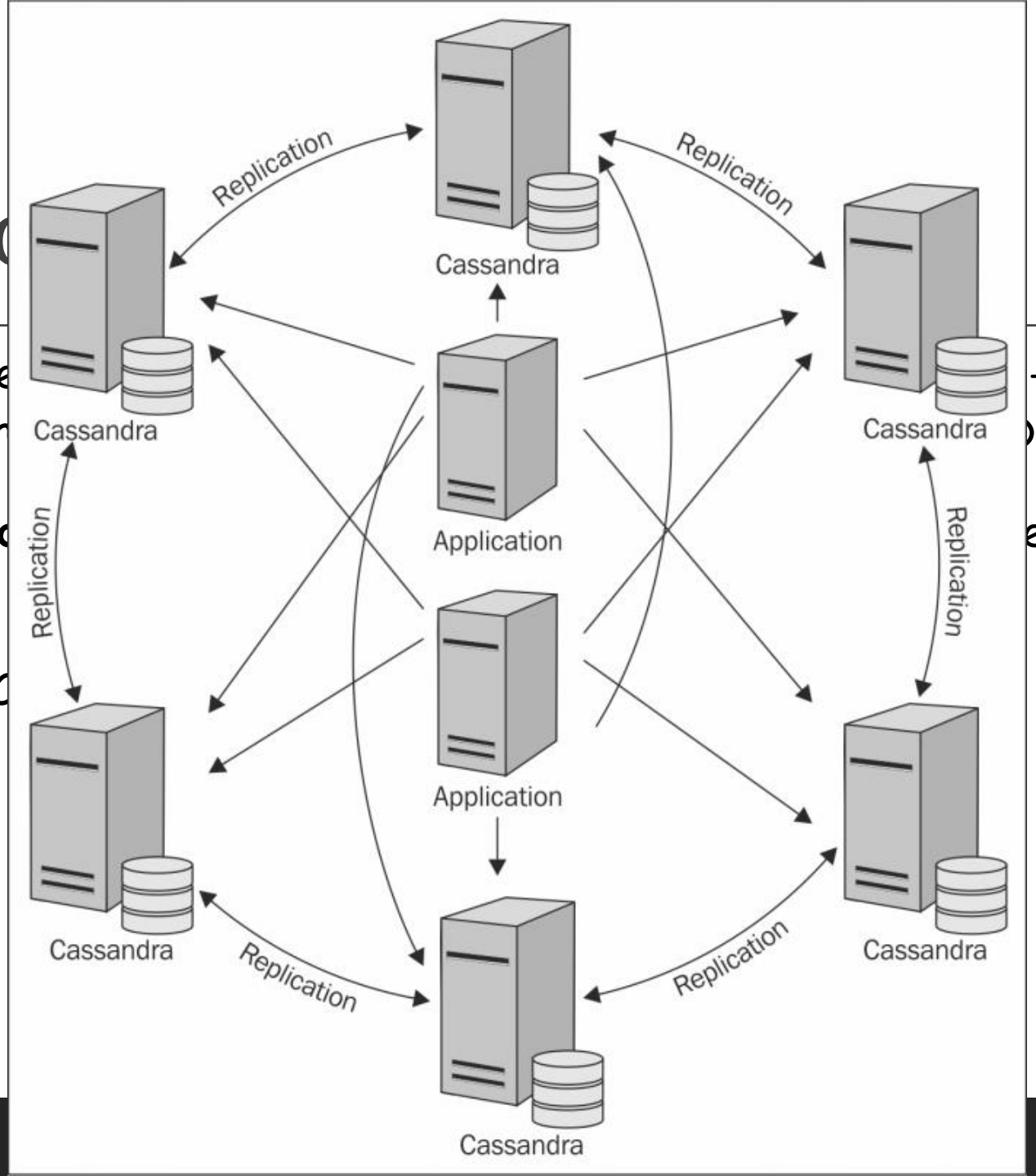
Client/ :ture



Client/Server Reference Architecture

Peer-To

- After a decade -peer have made a comeback in th BMSs.
- **Apache Casand** eer DDBMS and makes use of an
- All nodes in a C



Distributed query processing

- Distributed query processing is the process of executing a database query that involves data stored on multiple nodes or servers in a distributed database system.
- When a query is submitted, it must be broken down into smaller subqueries that can be executed on different nodes in parallel.
- The results must be combined to form the final result set.
- Distributed query processing involves several steps, including query **optimization**, query **decomposition**, data **fragmentation** and **distribution**, data **transfer**, **local processing**, and **result consolidation**.

Distributed query processing

The goal of distributed query processing is to minimize the amount of data that needs to be transferred between nodes and to maximize parallelism in the execution of subqueries in order to improve query performance .

Query processing problem

- The main function of a **relational query processor** is to transform a high-level query (typically, in relational calculus) into an equivalent lower-level query (typically, in some variation of relational algebra).
- The low-level query actually implements the **execution strategy** for the query and The transformation must achieve both **correctness** and **efficiency**.

Distributed query processing

Query processing problem

- The main function of a **relational query processor** is to transform a high-level query (typically, in relational calculus) into an equivalent lower-level query (typically, in some variation of relational algebra).
- The low-level query actually implements the execution strategy for the query and The transformation must achieve both **correctness** and **efficiency**.

Since each equivalent execution strategy can lead to very different consumptions of computer resources, the main difficulty is to select the execution strategy that minimizes resource consumption.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. 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 Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Relational schema. Set of employees (**EMP**) assigned (**ASG**) to projects (**PROJ**) for different payments (**PAY**)

Distributed query processing

Query processing problem (Example)

following simple user query: "Find the names of employees who are managing a project".

```
EMP(ENO, ENAME, TITLE)
ASG(ENO, PNO, RESP, DUR)
```

The expression of the query in relational calculus using the SQL syntax is

```
SELECT ENAME
FROM EMP,ASG
WHERE EMP.ENO = ASG.ENO
AND RESP = "Manager"
```

Distributed query processing

Query processing problem (Example 1)

Two equivalent relational algebra queries that are correct transformations of the query above are:

$$\Pi_{\text{NAME}}(\sigma_{\text{RESP}=\text{"Manager"} \wedge \text{EMP.ENO}}(\text{EMP} \times \text{ASG})), \text{ and}$$
$$\Pi_{\text{NAME}}(\text{EMP} \bowtie_{\text{ENO}} (\sigma_{\text{RESP}=\text{"Manager"}}(\text{ASG})))$$

It is intuitively obvious that the second query, which avoids the Cartesian product of EMP and ASG, consumes much less computing resources than the first, and thus should be retained.

Distributed query processing

Query processing problem

- In a centralized context, query execution strategies can be well expressed in an extension of relational algebra
- The main role of a centralized query processor is to choose, for a given query, the best relational algebra query among all equivalent ones.
- In a distributed system, relational algebra is not enough to express execution strategies. It must be supplemented with operators for exchanging data between sites
- In addition to the relational algebra operators, the distributed query processor must also select the best sites to process data, and possibly the way data should be transformed.

Distributed query processing

Query processing problem (Example 2)

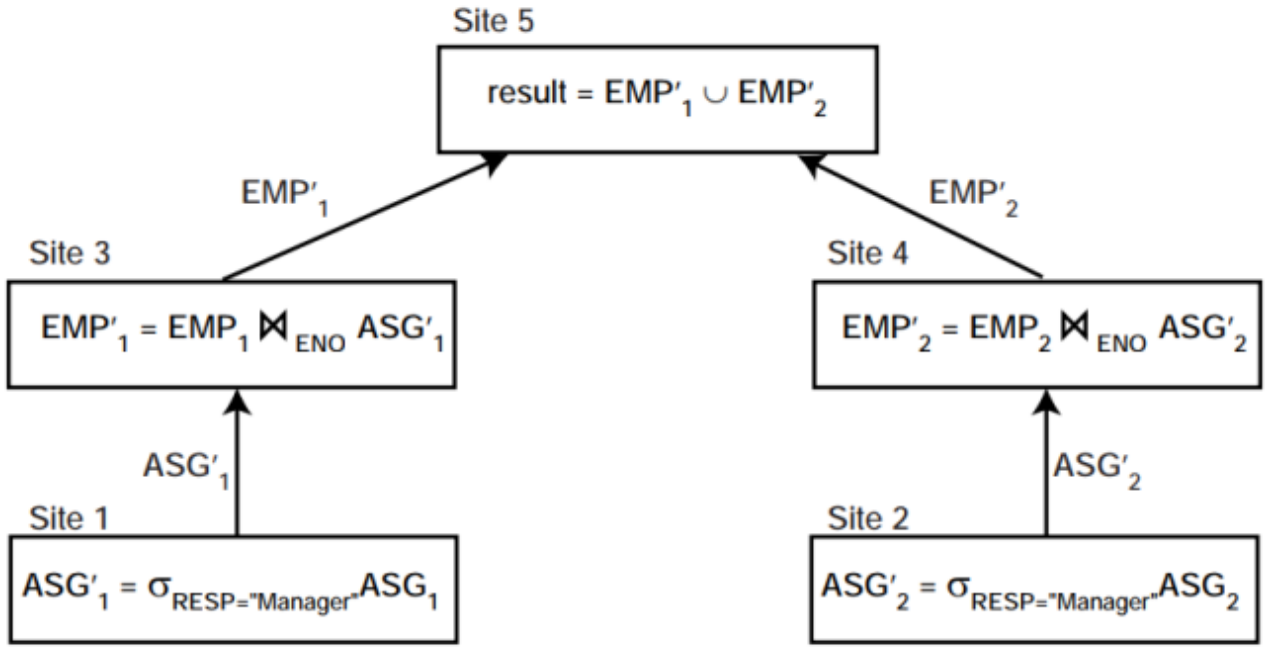
- We consider the following query

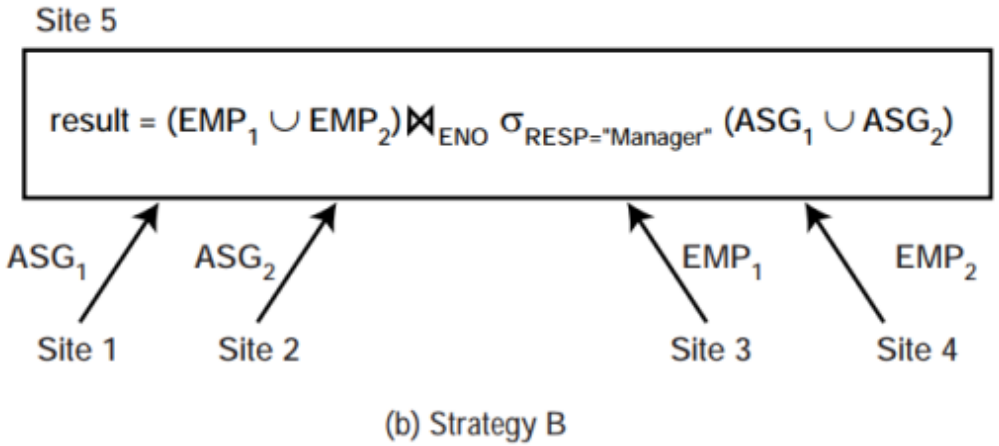
$$\Pi_{\text{NAME}}(\text{EMP} \bowtie_{\text{ENO}} (\sigma_{\text{RESP}=\text{"Manager"}}(\text{ASG})))$$

- We assume that relations EMP and ASG are horizontally fragmented as follows

$$\begin{aligned}
\text{EMP}_1 &= \sigma_{\text{ENO} \leqslant \text{"E3"}}(\text{EMP}) \\
\text{EMP}_2 &= \sigma_{\text{ENO} > \text{"E3"}}(\text{EMP}) \\
\text{ASG}_1 &= \sigma_{\text{ENO} \leqslant \text{"E3"}}(\text{ASG}) \\
\text{ASG}_2 &= \sigma_{\text{ENO} > \text{"E3"}}(\text{ASG})
\end{aligned}$$

Distributed query processing





(b) Strategy B

Distributed database design

In the design of a distributed DBMSs, the distribution of applications involves two things

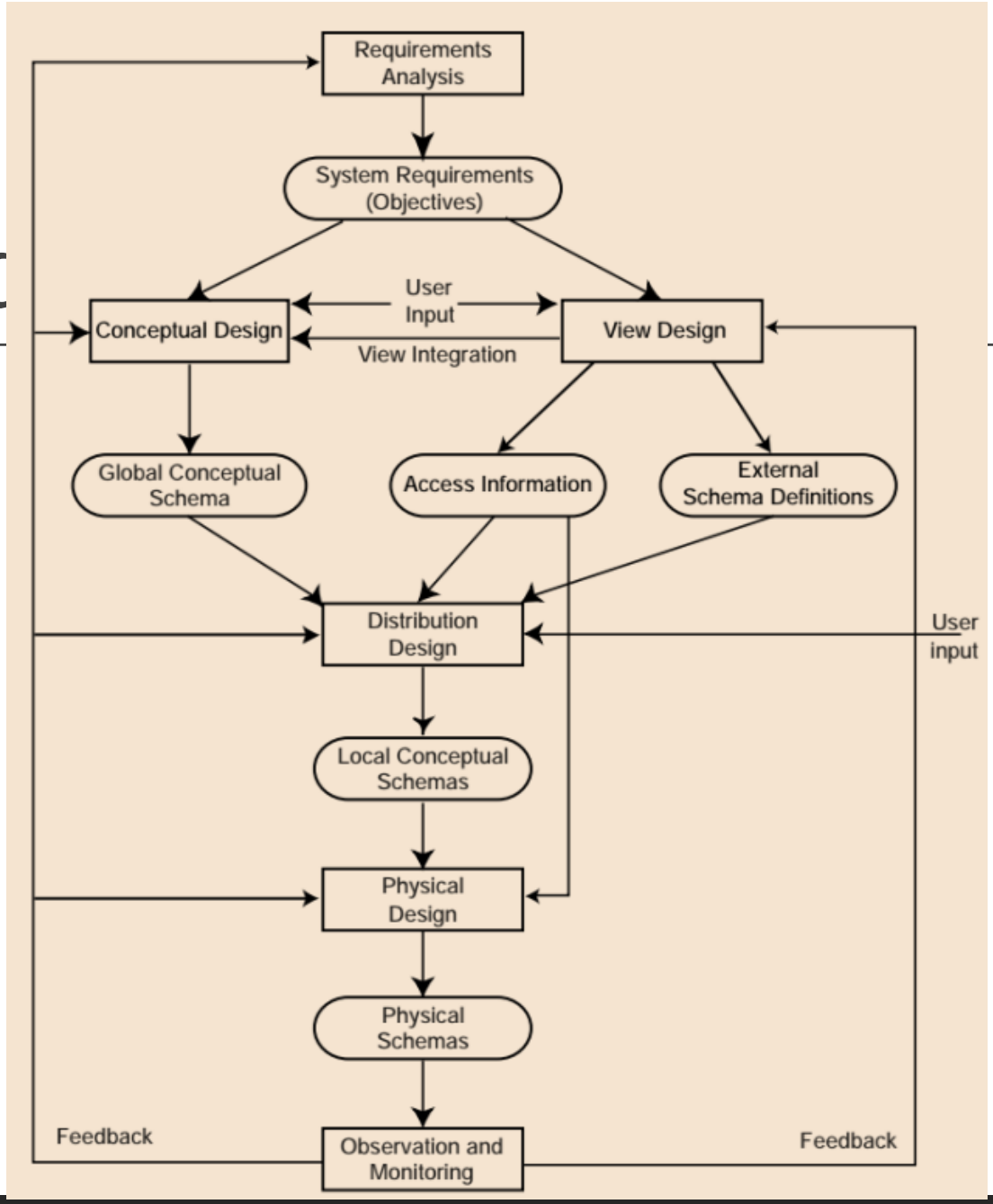
- The distribution of the distributed DBMS software, and
- The distribution of the application programs that run on it

Two major strategies that have been identified for designing distributed databases

The **top-down** approach and the **bottom-up** approach

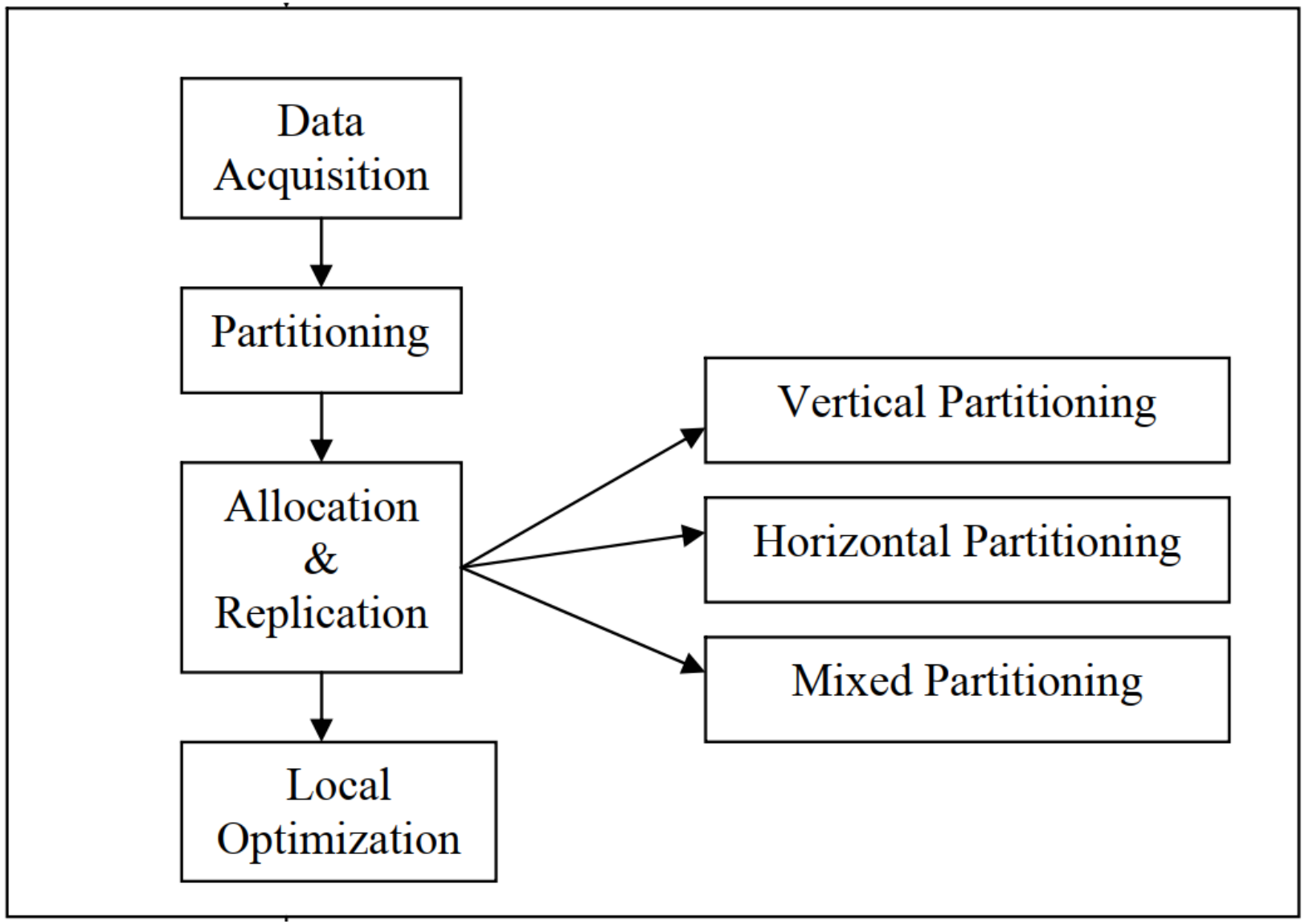
Distributed

Top-down approach



Dist

Distributio



Fragr

Vertical and horiz

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. 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 Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Correctness Rules of Fragmentation

Completeness

If a relation instance R is decomposed into fragments $F_R = \{R_1, R_2, ..., R_n\}$, each data item that can be found in R can also be found in one or more of R_i's. This property, which is identical to the *lossless* decomposition property of normalization, is also important in fragmentation since it ensures that the data in a global relation are mapped into fragments without any loss. Note that in the case of horizontal fragmentation, the "item" typically refers to a tuple, while in the case of vertical fragmentation, it refers to an attribute.

Correctness Rules of Fragmentation

Reconstruction

If a relation R is decomposed into fragments $F_R = \{R_1, R_2, ..., R_n\}$, it should be possible to define a relational operator ∇ such that $R = \nabla R_i, \forall R_i \in F_R$. The operator ∇ will be different for different forms of fragmentation; it is important, however, that it can be identified. The reconstructability of the relation from its fragments ensures that constraints defined on the data in the form of dependencies are preserved.

Correctness Rules of Fragmentation

Disjointness

If a relation R is horizontally decomposed into fragments $F_R = \{R_1, R_2, ..., R_n\}$ and data item d_i is in R_j, it is not in any other fragment $R_k (k \neq j)$. This criterion ensures that the horizontal fragments are disjoint. If relation R is vertically decomposed, its primary key attributes are typically repeated in all its fragments (for reconstruction). Therefore, in case of vertical partitioning, disjointness is defined only on the non-primary key attributes of a relation.