SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCSA3008

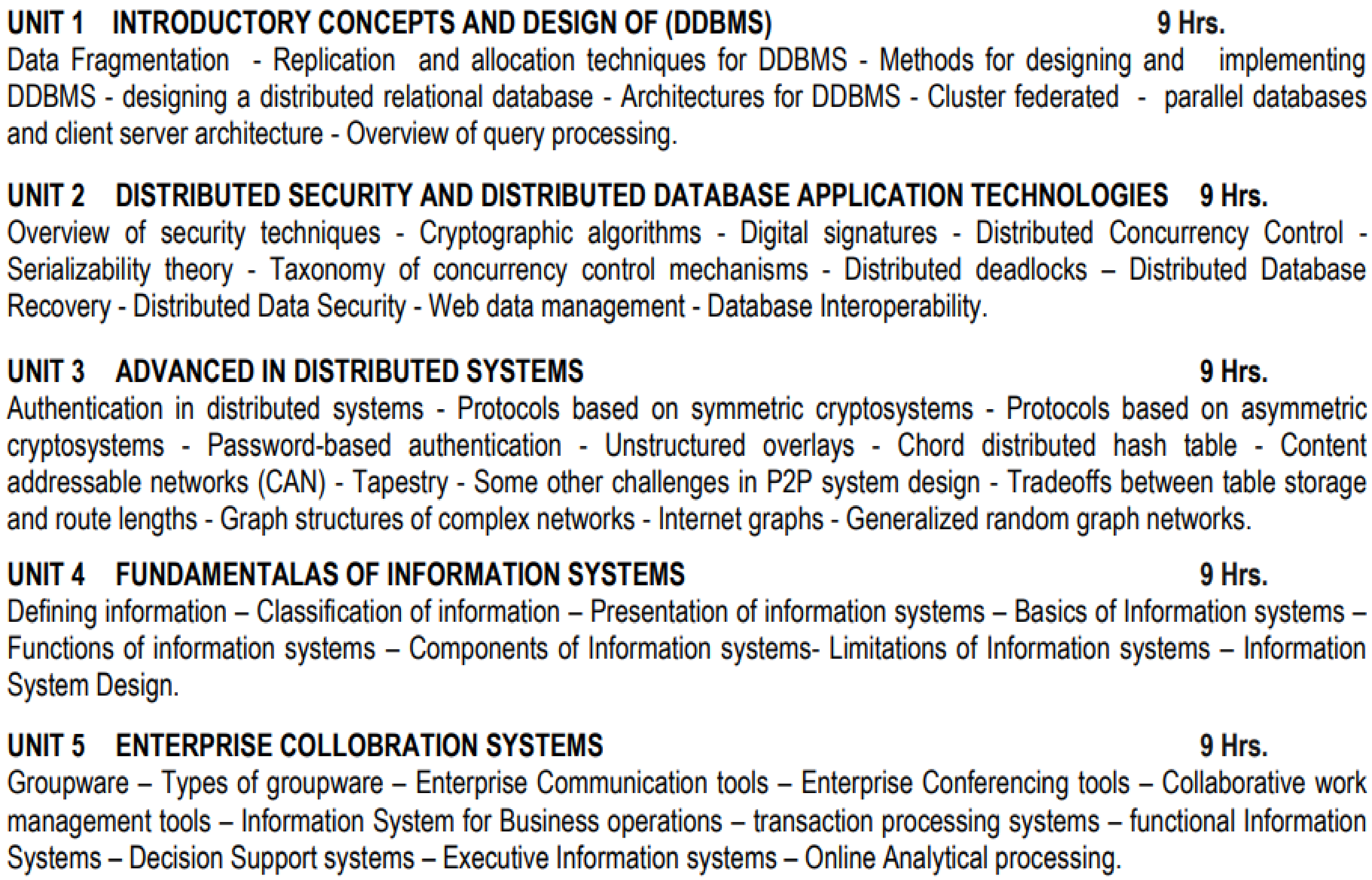
DISTRIBUTED DATABASE AND INFORMATION SYSTEMS

Dr.R.Sathya Bama Krishna,
Associate Professor,
Dept.of CSE

COURSE OBJECTIVES

- To understand the role of databases and database management systems in managing organizational data and information.
- To understand the techniques used for data fragmentation, replication and allocation during the distributed database design process.
- To discuss the issues involved in resource management and process.
- To Perceive the building blocks and design of information systems.
- To acquire knowledge of information systems on Business operations

SYLLABUS



UNIT 1 INTRODUCTORY CONCEPTS AND DESIGN OF (DDBMS) 9 Hrs.

Data Fragmentation - Replication and allocation techniques for DDBMS - Methods for designing and implementing DDBMS - designing a distributed relational database - Architectures for DDBMS - Cluster federated - parallel databases and client server architecture - Overview of query processing.

UNIT 2 DISTRIBUTED SECURITY AND DISTRIBUTED DATABASE APPLICATION TECHNOLOGIES 9 Hrs.

Overview of security techniques - Cryptographic algorithms - Digital signatures - Distributed Concurrency Control - Serializability theory - Taxonomy of concurrency control mechanisms - Distributed deadlocks – Distributed Database Recovery - Distributed Data Security - Web data management - Database Interoperability.

UNIT 3 ADVANCED IN DISTRIBUTED SYSTEMS 9 Hrs.

Authentication in distributed systems - Protocols based on symmetric cryptosystems - Protocols based on asymmetric cryptosystems - Password-based authentication - Unstructured overlays - Chord distributed hash table - Content addressable networks (CAN) - Tapestry - Some other challenges in P2P system design - Tradeoffs between table storage and route lengths - Graph structures of complex networks - Internet graphs - Generalized random graph networks.

UNIT 4 FUNDAMENTALAS OF INFORMATION SYSTEMS 9 Hrs.

Defining information – Classification of information – Presentation of information systems – Basics of Information systems – Functions of information systems – Components of Information systems- Limitations of Information systems – Information System Design.

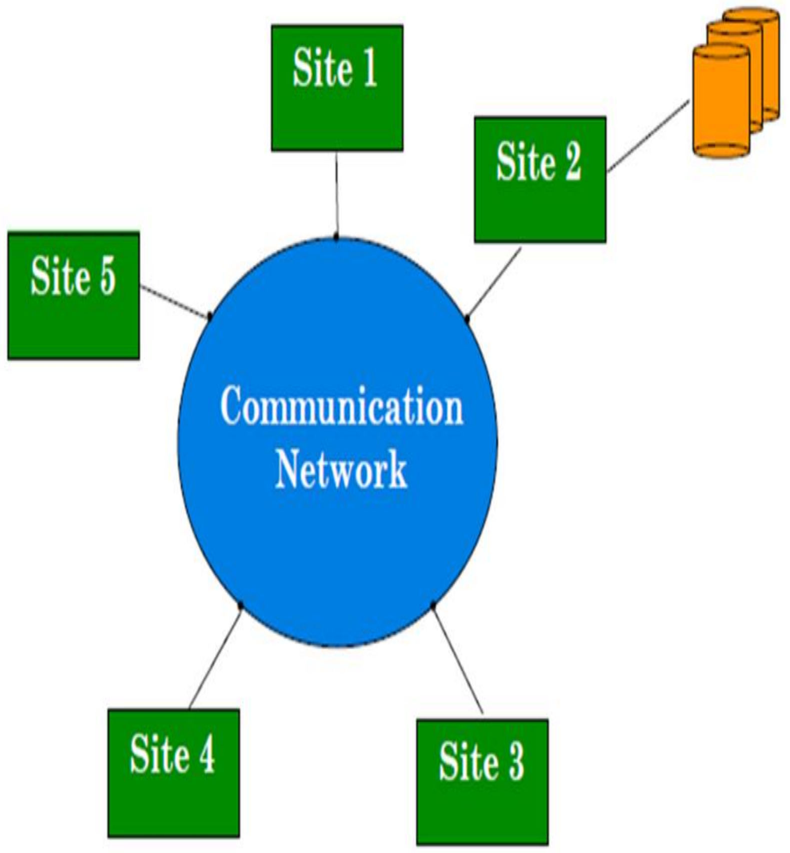
UNIT 5 ENTERPRISE COLLOBRATION SYSTEMS 9 Hrs.

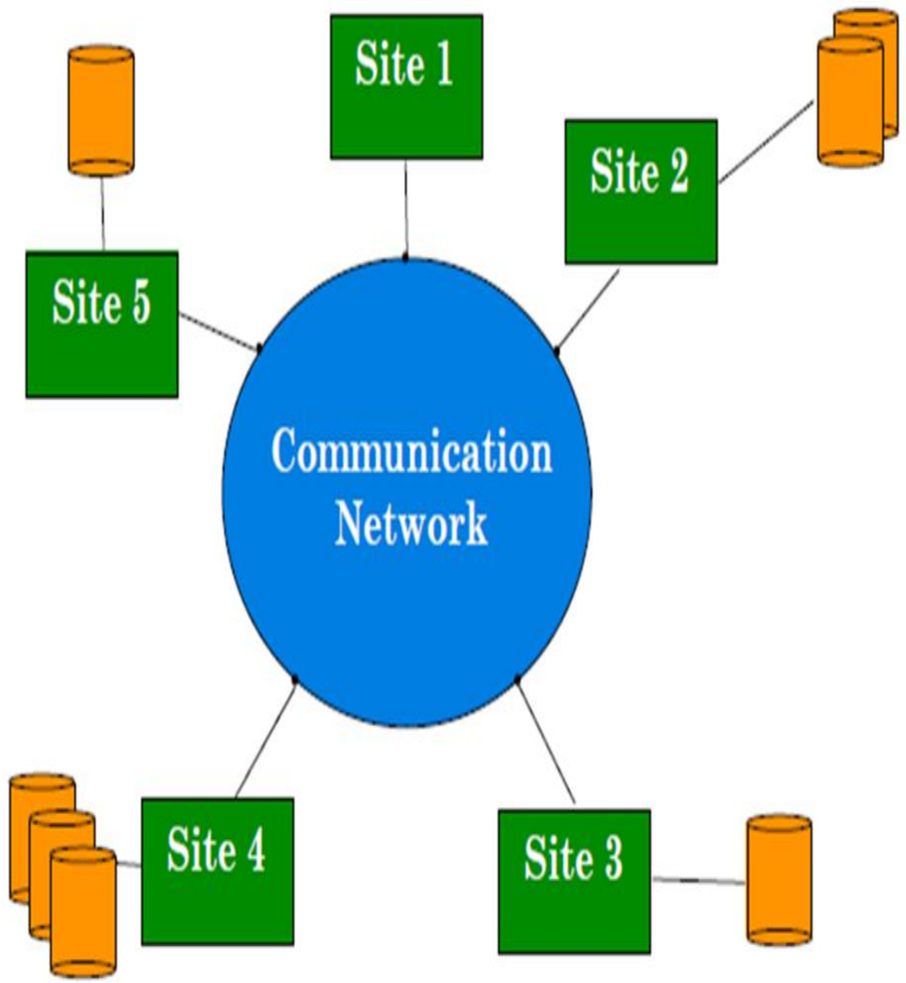
Groupware – Types of groupware – Enterprise Communication tools – Enterprise Conferencing tools – Collaborative work management tools – Information System for Business operations – transaction processing systems – functional Information Systems – Decision Support systems – Executive Information systems – Online Analytical processing.

Expected Background

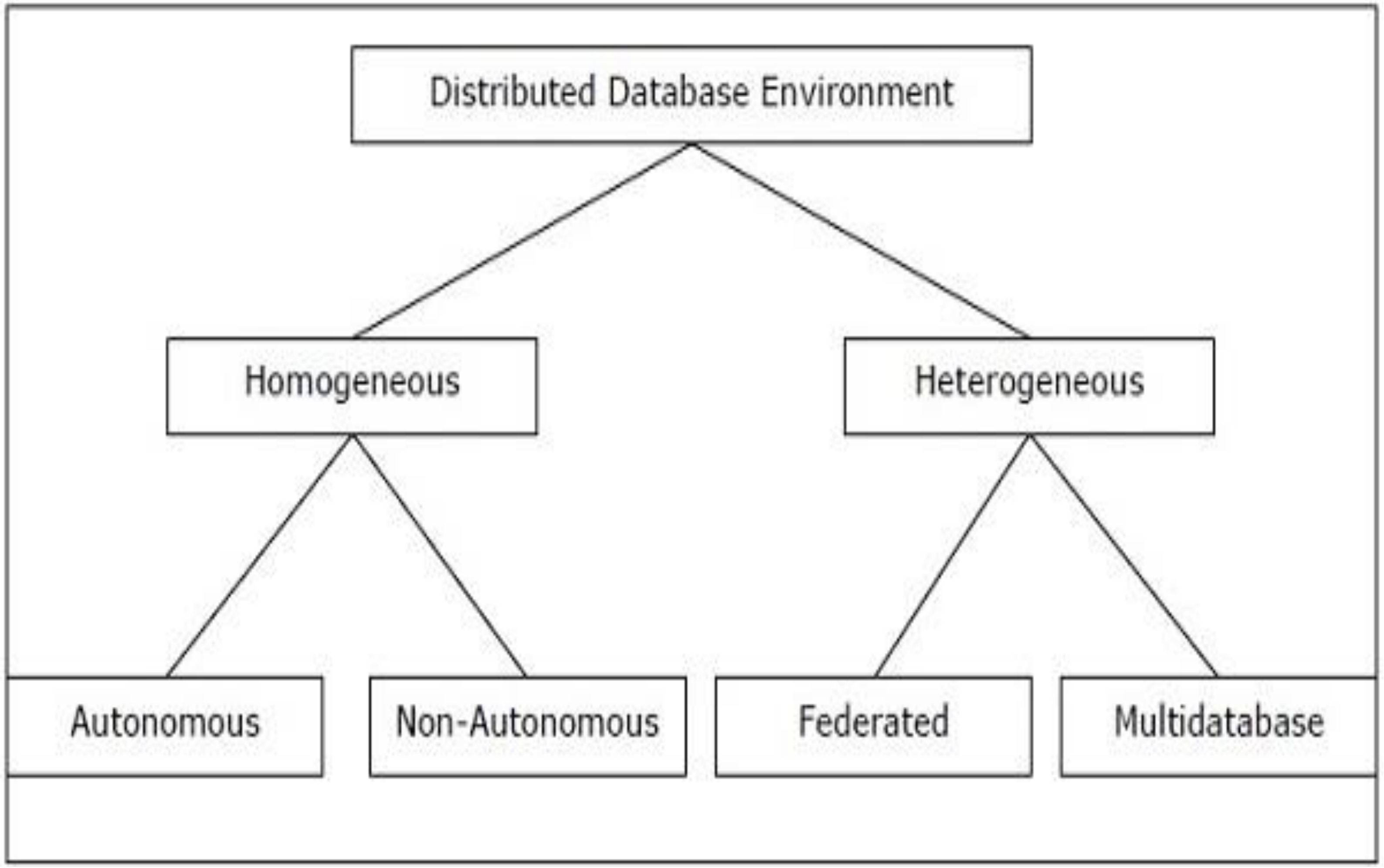
- Basic SQL
- Relational algebra
- Following aspects of centralized DB
  - Query processing: query plans, cost estimation, optimization
  - Concurrency control techniques
  - Recovery methods

DDBMS


Site 1
Site 2
Site 5
Communication Network
Site 4
Site 3


Site 1
Site 2
Site 5
Communication Network
Site 4
Site 3

Types of Distributed Databases



Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters :

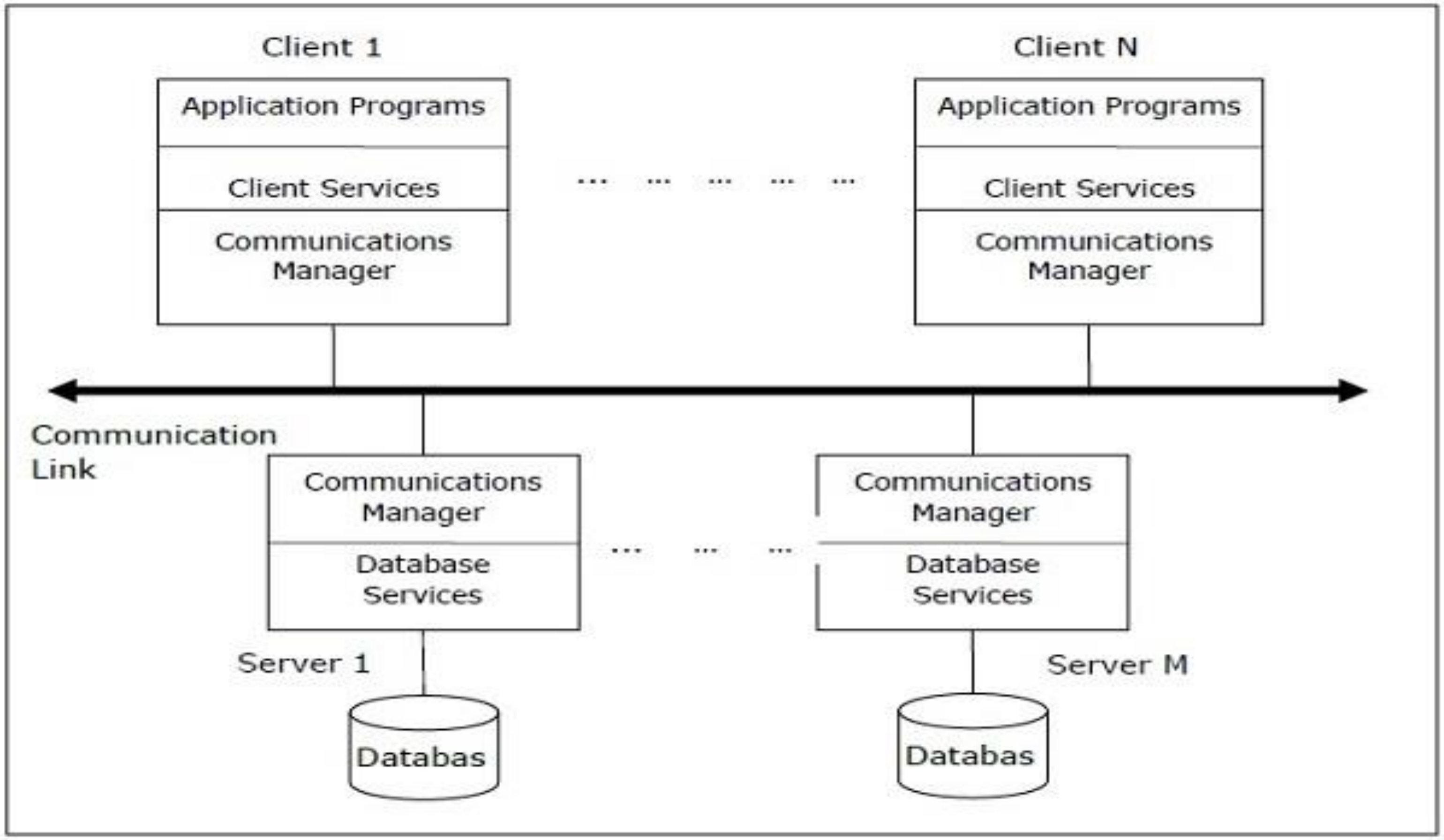
- **Distribution** – It states the physical distribution of data across the different sites.
- **Autonomy** – It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
- **Heterogeneity** – It refers to the uniformity or dissimilarity of the data models, system components and databases.

Architectural Models

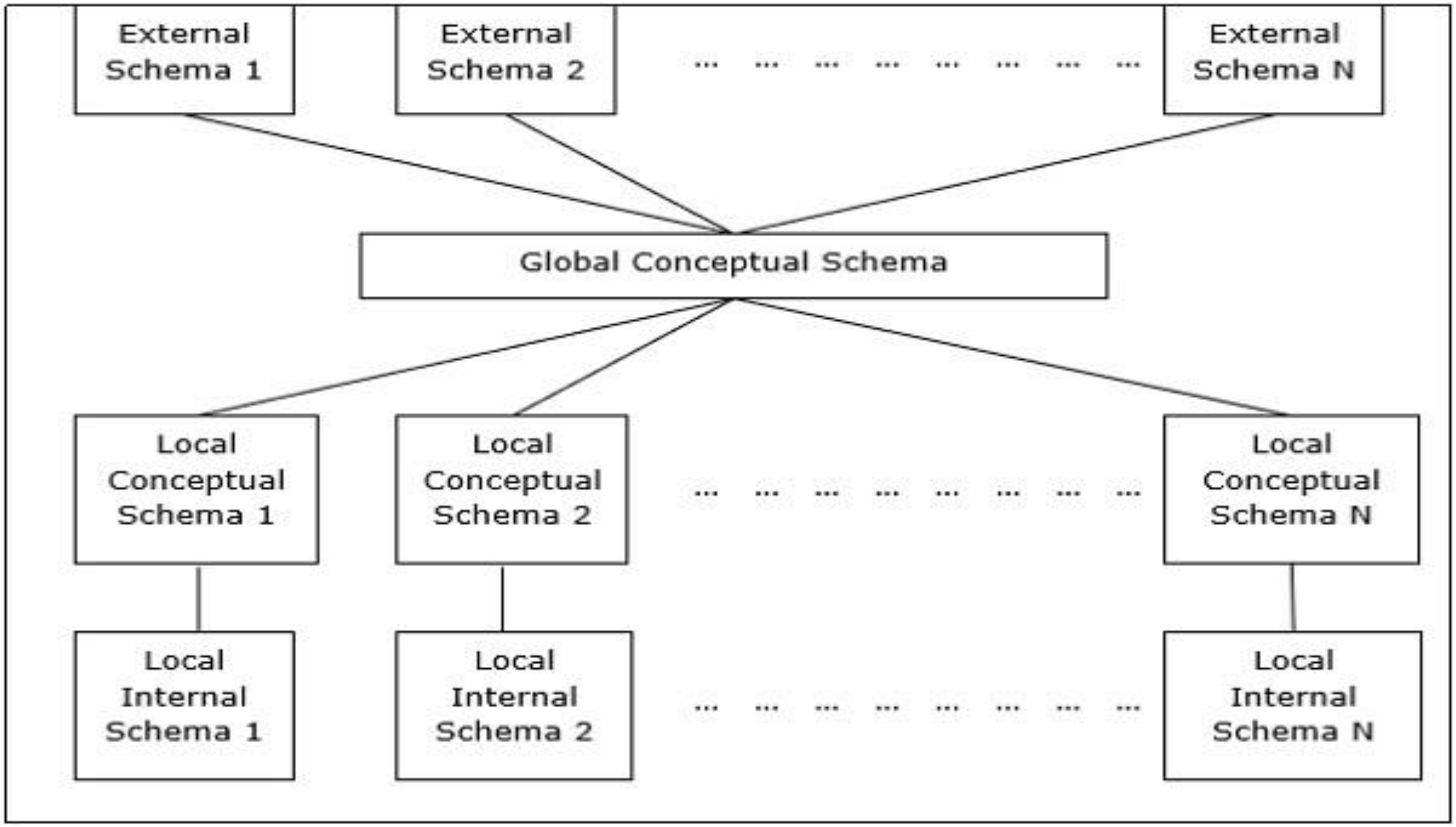
Some of the common architectural models are :

- Client - Server Architecture for DDBMS
- Peer - to - Peer Architecture for DDBMS
- Multi - DBMS Architecture

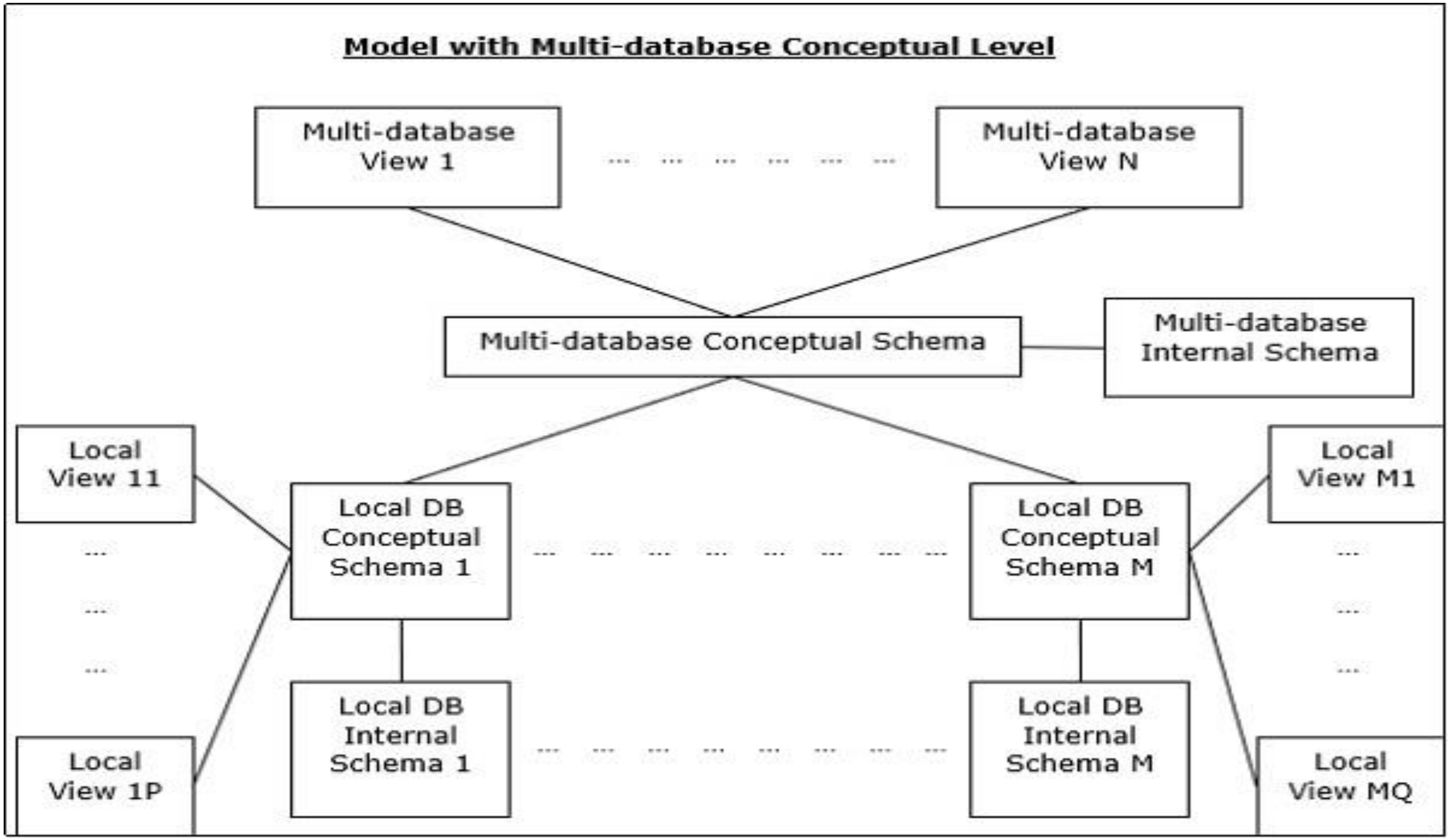
Client - Server Architecture for DDBMS



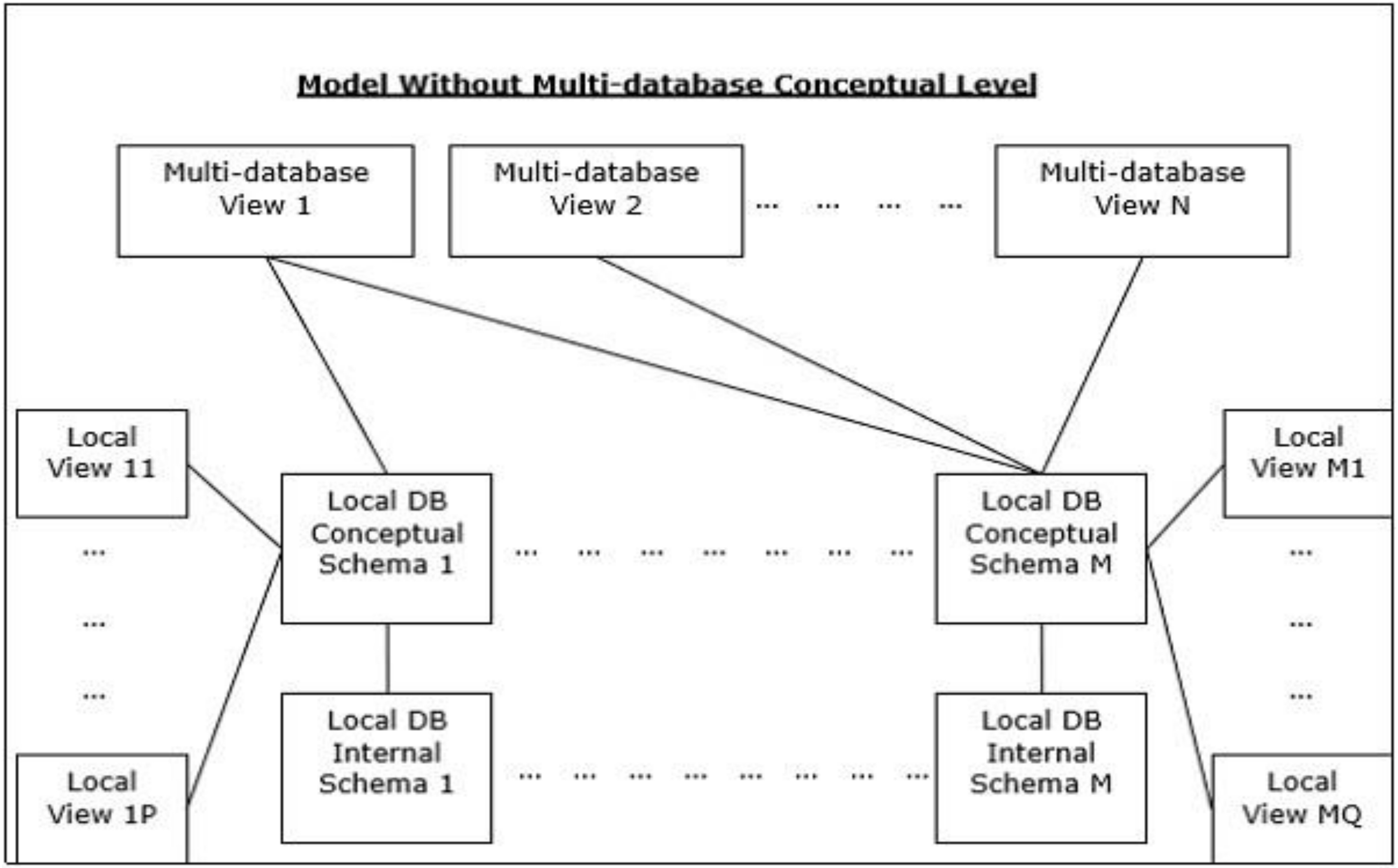
Peer- to-Peer Architecture for DDBMS



Multi - DBMS Architectures


Model with Multi-database Conceptual Level
Multi-database View 1
Multi-database View N
Multi-database Conceptual Schema
Multi-database Internal Schema
Local View 11
Local View 1P
Local DB Conceptual Schema 1
Local DB Internal Schema 1
Local DB Conceptual Schema M
Local DB Internal Schema M
Local View M1
Local View MQ

Multi - DBMS Architectures



Design Alternatives

The distribution design alternatives for the tables in a DDBMS are as follows :

- Non-replicated and non-fragmented
- Fully replicated
- Partially replicated
- Fragmented
- Mixed

Data Replication

- Data replication is the process of storing separate copies of the database at two or more sites.
- It is a popular fault tolerance technique of distributed databases.

Advantages and Disadvantages

Advantages :

- Reliability
- Reduction in Network Load
- Quicker Response
- Simpler Transactions

Disadvantages :

- Increased Storage Requirements
- Increased Cost and Complexity of Data Updating
- Undesirable Application–Database coupling

Fragmentation

- Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**.
- Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical).
- Fragmentation should be done in a way so that the original table can be reconstructed from the fragments.

Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields.

Horizontal fragmentation should also confirm to the rule of reconstructiveness.

```
CREATE COMP_STD AS
   SELECT * FROM STUDENT
   WHERE COURSE = "Computer Science";
```

Horizontal Fragmentation

This EMPLOYEE table can be divided into different fragments like:

EMP 1 = $\sigma_{Dep = 1}$ EMPLOYEE

EMP 2 = $\sigma_{Dep = 2}$ EMPLOYEE

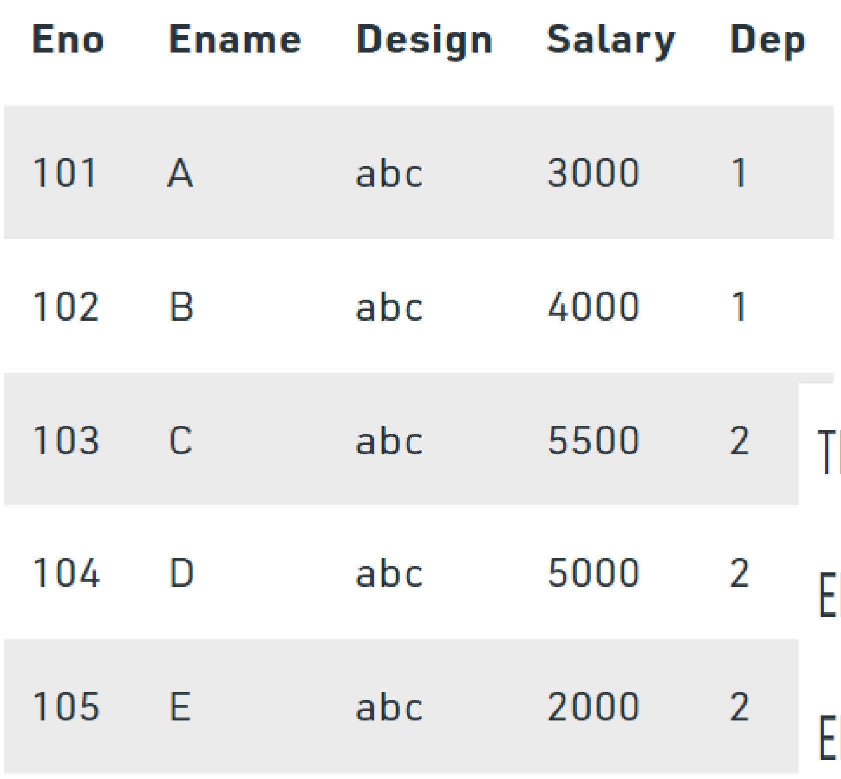
These two fragments are: T1 fragment of Dep = 1

Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1

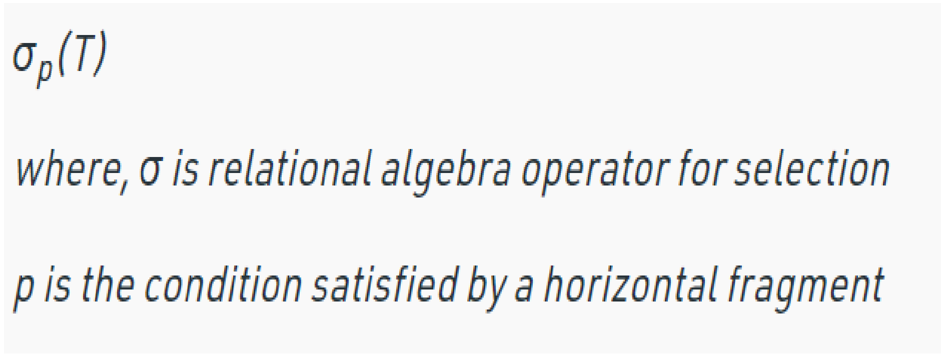
Eno	Ename	Design	Salary	Dep
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2

Horizontal Fragmentation

For example, consider an EMPLOYEE table (T) :



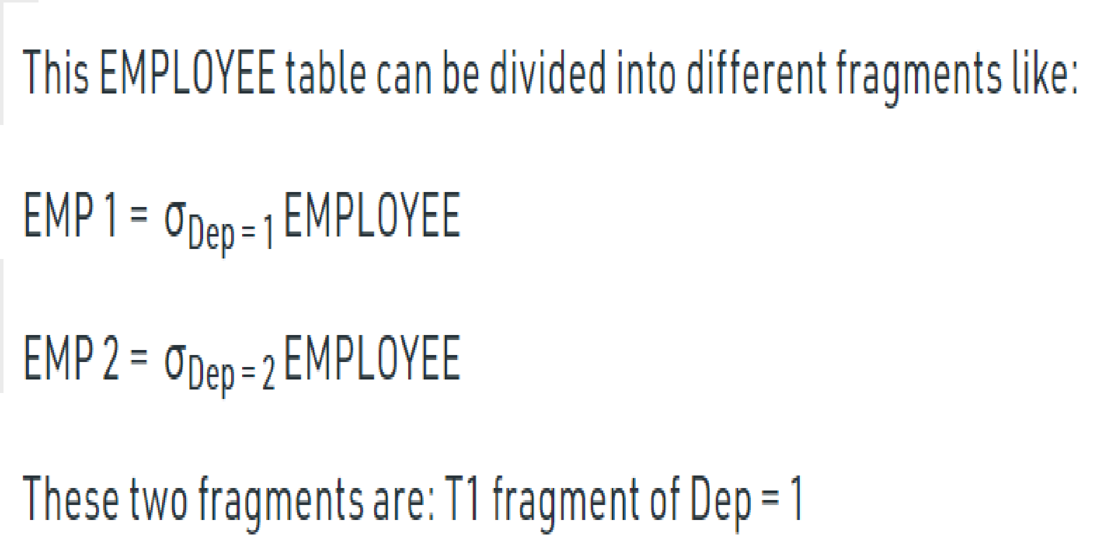
Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2



$\sigma_p(T)$

where, σ is relational algebra operator for selection

p is the condition satisfied by a horizontal fragment



This EMPLOYEE table can be divided into different fragments like:

EMP 1 = $\sigma_{Dep = 1}$ EMPLOYEE

EMP 2 = $\sigma_{Dep = 2}$ EMPLOYEE

These two fragments are: T1 fragment of Dep = 1

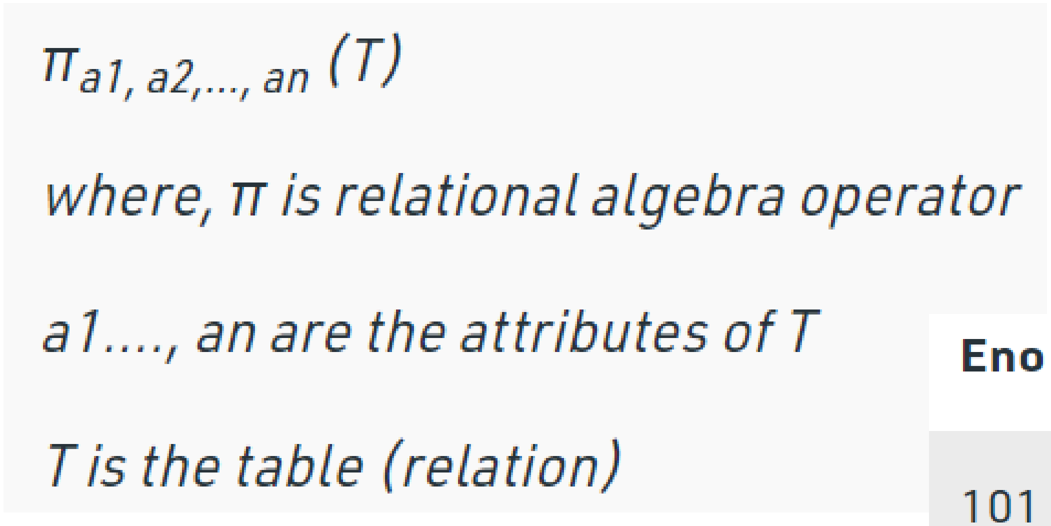
Vertical Fragmentation

STUDENT

Regd_No	Name	Course	Address	Semester	Fees	Marks

```
CREATE TABLE STD_FEES AS
    SELECT Regd_No, Fees
    FROM STUDENT;
```

Vertical Fragmentation

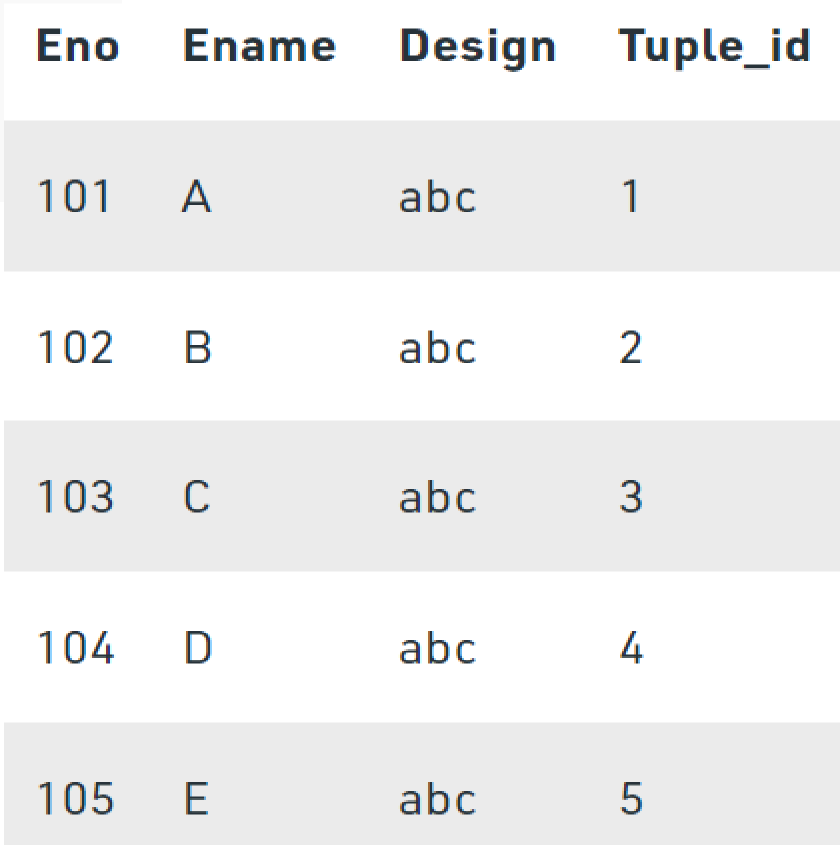


$\pi_{a1,\, a2,\ldots,\, an}\,(T)$

where, π is relational algebra operator

a1...., an are the attributes of T

T is the table (relation)



Eno	Ename	Design	Tuple_id
101	A	abc	1
102	B	abc	2
103	C	abc	3
104	D	abc	4
105	E	abc	5

Vertical Fragmentation

$\pi_{a1,\, a2, \ldots,\, an}\ (T)$

where, π is relational algebra operator

a1…., an are the attributes of T

T is the table (relation)

Salary	Dep	Tuple_id
3000	1	1
4000	2	2
5500	3	3
5000	1	4
2000	4	5

Eno	Ename	Design	Tuple_id
101	A	abc	1
102	B	abc	2
103	C	abc	3
104	D	abc	4
105	E	abc	5

Hybrid Fragmentation

Hybrid fragmentation can be done in two ways :

- At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.

- At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

Employee Table

WORKER_ ID	FIRST_ NAME	LAST_NAM E	SALARY	JOINING_DATE	DEPARTMENT
001	Monika	Arora	100000	2014-02-20 09:00:00	HR
002	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
003	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
004	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
005	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
006	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
007	Satish	Kumar	75000	2014-01-20 09:00:00	Account
008	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

Bonus -Table

WORKER_ REF_ID	BONUS_DATE	BONUS_ AMOUNT
1	2016-02-20 00:00:00	5000
2	2016-06-11 00:00:00	3000
3	2016-02-20 00:00:00	4000
1	2016-02-20 00:00:00	4500
2	2016-06-11 00:00:00	3500

Title – Table

WORKER_ REF_ID	WORKER_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

Write an SQL query to fetch “FIRST_NAME” from Employee table using the alias name as <EMPLOYEE_NAME>.

Select FIRST_NAME AS EMPLOYEE_NAME from Employee;

Write an SQL query to fetch “FIRST_NAME” from Employee table in upper case.

Select upper (FIRST_NAME) from Employee;

Write an SQL query to fetch unique values of DEPARTMENT from Employee table.

Select distinct DEPARTMENT from Employee;

Write an SQL query to print the first three characters of FIRST_NAME from Employee table.

Select substring (FIRST_NAME,1,3) from Employee;

Write an SQL query to fetch departments along with the total salaries paid for each of them.

SELECT DEPARTMENT, sum(Salary) from worker group by DEPARTMENT;

Write an SQL query to fetch the names of workers who earn the highest salary.

SELECT FIRST_NAME, SALARY from Worker WHERE SALARY=(SELECT max(SALARY) from Worker);

Write an SQL query to find the position of the alphabet (‘a’) in the first name column ‘Amitabh’ from Employee table.

Select INSTR(FIRST_NAME, BINARY'a') from Worker where FIRST_NAME = 'Amitabh';

Write an SQL query to print the FIRST_NAME from Worker table after replacing ‘a’ with ‘A’.

```
Select REPLACE(FIRST_NAME,'a','A') from Worker;
```

Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending.

```
Select * from Worker order by FIRST_NAME asc;
```

Write an SQL query to print details of the Workers whose FIRST_NAME contains ‘a’.

```
Select * from Worker where FIRST_NAME like '%a%';
```

Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000.

Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000.

```
SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) As
Worker_Name, Salary
FROM worker
WHERE WORKER_ID IN
(SELECT WORKER_ID FROM worker
WHERE Salary BETWEEN 50000 AND 100000);
```

Cluster Federated Databases

- A federated database system (FDBS) is a type of meta-database management system (DBMS), which transparently maps multiple autonomous database systems into a single federated database.
- The constituent databases are interconnected via a computer network and may be geographically decentralized.
- Since the constituent database systems remain autonomous, a federated database system is a contrastable alternative to the task of merging several disparate databases.
- A federated database, or virtual database, is a composite of all constituent databases in a federated database system.
- There is no actual data integration in the constituent disparate databases as a result of data federation.

Federated Databases

- A multiple DBS (MDBS) can be classified into two types depending on the autonomy of the component DBS as Federated and Non Federated.
- A Nonfederated database system is an integration of component DBMS that are not autonomous.
- A Federated Database system consists of component DBS that are autonomous yet participate in a federation to allow partial and controlled sharing of their data.
- Federated architectures differ based on levels of integration with the component database systems and the extent of services offered by the federation.
- A FDBS can be categorized as :

  Loosely or Tightly Coupled Systems.

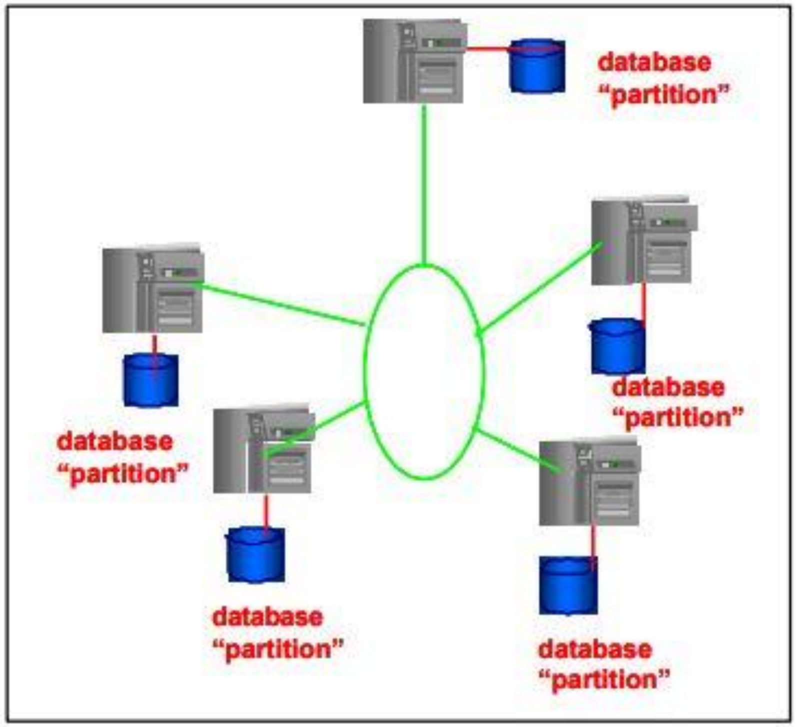
Federated Databases

- Loosely Coupled require component databases to construct their own federated schema. A user will typically access other component database systems by using a multidatabase language but this removes any levels of location transparency, forcing the user to have direct knowledge of the federated schema. A user imports the data they require from other component databases and integrates it with their own to form a federated schema.

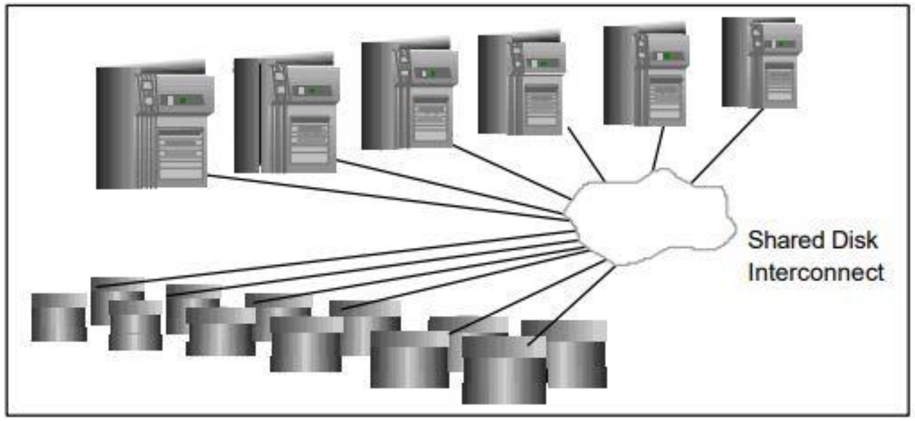
- Tightly coupled system consists of component systems that use independent processes to construct and publicize an integrated federated schema.

Cluster Federated Databases

- Multiple DBS of which FDBS are a specific type can be characterized along three dimensions: Distribution, Heterogeneity and Autonomy.



Federated DBMS



Cluster Federated DBMS

Cluster Federated Databases

- A cluster consists of servers, a cluster interconnect and a shared disk subsystem. Shared disk database architectures run on hardware clusters that give every participating server equal access to all disks – however, servers do not share memory. Most major hardware vendors provide shareddisk clusters today.
- A database instance runs on every node of the cluster. Transactions running on any instance can read or update any part of the database there is no notion of data ownership by a node.
- System performance is based on the database effectively utilizing a fast interconnect, such as the Virtual Interface Architecture (VIA), between cluster nodes.
- Oracle9i Real Application Clusters (RAC) is the first successful shared-disk cluster architecture and utilizes sophisticated Cache Fusion shared-cache algorithms to allow high performance and scalability without data or application partitioning.

Parallel Databases

Parallel Databases:

This is a database system running on a parallel computer.

Why parallel DBs?

To make processing faster. Intuitively place different parts of a large table on different processors and perform queries in parallel.

3 Types:

1. Shared Memory Parallel Databases Several processors share RAM memory and also disks.
2. Shared Disk Parallel Databases Each processor has its own memory, but they share the disks.
3. Shared Nothing Parallel Databases Each processor has its own memory and its own disk(s).

Parallel Databases

- The processors communicate through a high-speed network. Due to the increased speed of Local Area Networks (LANs) and Wide Area Networks (WANs) Shared Nothing parallel databases have become very similar to distributed databases.
- The main difference is in parallel computers the use of components that are all the same. Same hardware, same system and same software and they are in one rack. Keep the distance between CPUs short, because long wires mean slow communication.
- One of the main parallel processing initiative in the US is IBM's Blue Gene computer for the protein folding problem.

Difference between Client/Server and Distributed DBMS :

1.	Client can access only one server at a time.	User can access many sites simultaneously.
2.	It is difficult to manage.	It is easy to manage.
3.	In this data is distributed across clients.	In this data is distributed across sites.
4.	Speed of accessing database is poor as compared to Distributed DBMS.	Speed of accessing database is much better than Client/Server Architecture.
5.	If somehow server crashes, the whole system stops.	The crash of one site does not stop the entire system.
6.	Accessing of data is easy to control.	Accessing of data is difficult to control.
7.	It is less expensive as compared to Distributed DBMS.	It is expensive.
8.	Maintenance cost is low.	Maintenance cost is high.

Overview of Query Processing

- A Query processing in a distributed database management system requires the transmission of data between the computers in a network.
- A distribution strategy for a query is the ordering of data transmissions and local data processing in a database system.
- Generally, a query in Distributed DBMS requires data from multiple sites, and this need for data from different sites is called the transmission of data that causes communication costs.
- Query processing in DBMS is different from query processing in centralized DBMS due to this communication cost of data transfer over the network.
- The transmission cost is low when sites are connected through high-speed Networks and is quite significant in other networks.

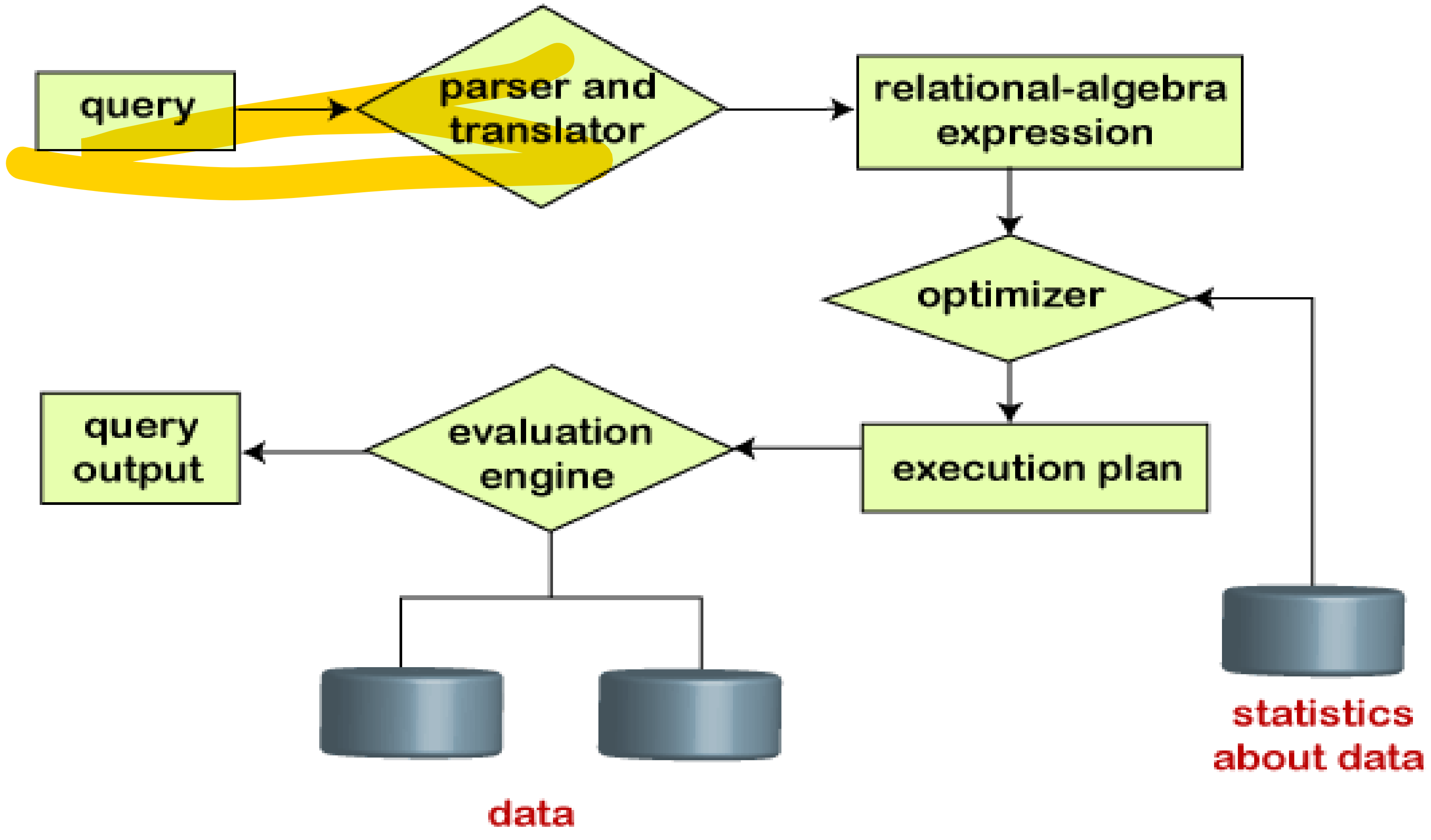
Query Processing

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database.

The steps involved are:

1. Parsing and translation
2. Optimization
3. Evaluation

Query Processing in DBMS



Steps in query processing

Communication Cost of DQP

- In Distributed Query processing, the data transfer cost means the cost of transferring intermediate files to other sites for processing and therefore the cost of transferring the ultimate result files to the location where that result's required.

- Let's say that a user sends a query to site S1, which requires data from its own and also from another site S2. Now, there are three strategies to process this query which are given below:

  - ➢ We can transfer the data from S2 to S1 and then process the query
  - ➢ We can transfer the data from S1 to S2 and then process the query
  - ➢ We can transfer the data from S1 and S2 to S3 and then process the query.

Communication Cost of DQP

- Here the choice depends on various factors like, the size of relations and the results, the communication cost between different sites, and at which the site result will be utilized.
- Commonly, the data transfer cost is calculated in terms of the size of the messages. By using the below formula, we can calculate the data transfer cost:

Data transfer cost = C * Size

where C refers to the cost per byte of data transferring and Size is the no. of bytes transmitted.

EXAMPLE

Consider the following table EMPLOYEE and DEPARTMENT.

Site1: EMPLOYEE

EID NAME SALARY DID

EID- 10 bytes
SALARY- 20 bytes
DID- 10 bytes
Name- 20 bytes
Total records- 1000
Record Size- 60 bytes

Site2: DEPARTMENT

DID DNAME

DID- 10 bytes
DName- 20 bytes
Total records- 50
Record Size- 30 bytes

EXAMPLE

Find the name of employees and their department names. Also, find the amount of data transfer to execute this query when the query is submitted to Site 3.

Solution : Considering the query is submitted at site 3 and neither of the two relations that is an EMPLOYEE and the DEPARTMENT not available at site 3. So, to execute this query, we have three strategies:

EXAMPLE

Transfer both the tables that is EMPLOYEE and DEPARTMENT at SITE 3 then join the tables there. The total cost in this case is :

1000 * 60 + 50 * 30 = 60,000 + 1500 = 61500 bytes.

Transfer the table EMPLOYEE to SITE 2, join the table at SITE 2 and then transfer the result at SITE 3. The total cost in this case is :

60 * 1000 + 60 * 1000 = 120000 bytes

Transfer the table DEPARTMENT to SITE 1, join the table at SITE 2 join the table at site1 and then transfer the result at site3. The total cost in this case is :

30 * 50 + 60 * 1000 = 61500 bytes

Using Semi join in Distributed Query processing

- The semi-join operation is used in distributed query processing to reduce the number of tuples in a table before transmitting it to another site.
- This reduction in the number of tuples reduces the number and the total size of the transmission that ultimately reducing the total cost of data transfer.
- Let's say that we have two tables R1, R2 on Site S1, and S2. Now, we will forward the joining column of one table say R1 to the site where the other table say R2 is located.
- This column is joined with R2 at that site. The decision whether to reduce R1 or R2 can only be made after comparing the advantages of reducing R1 with that of reducing R2.
- Thus, semi-join is a well-organized solution to reduce the transfer of data in distributed query processing.

EXAMPLE

Find the name of employees and their department names. Also, find the amount of data transfer to execute this query when the query is submitted to Site 3.

Find the amount of data transferred to execute the above query given in the above example using **using Semi join in Distributed Query processing.**

EXAMPLE

Consider the following table EMPLOYEE and DEPARTMENT.

Site1: EMPLOYEE

EID NAME SALARY DID

EID- 10 bytes
SALARY- 20 bytes
DID- 10 bytes
Name- 20 bytes
Total records- 1000
Record Size- 60 bytes

Site2: DEPARTMENT

DID DNAME

DID- 10 bytes
DName- 20 bytes
Total records- 50
Record Size- 30 bytes

EXAMPLE

Solution : The following strategy can be used to execute the query.

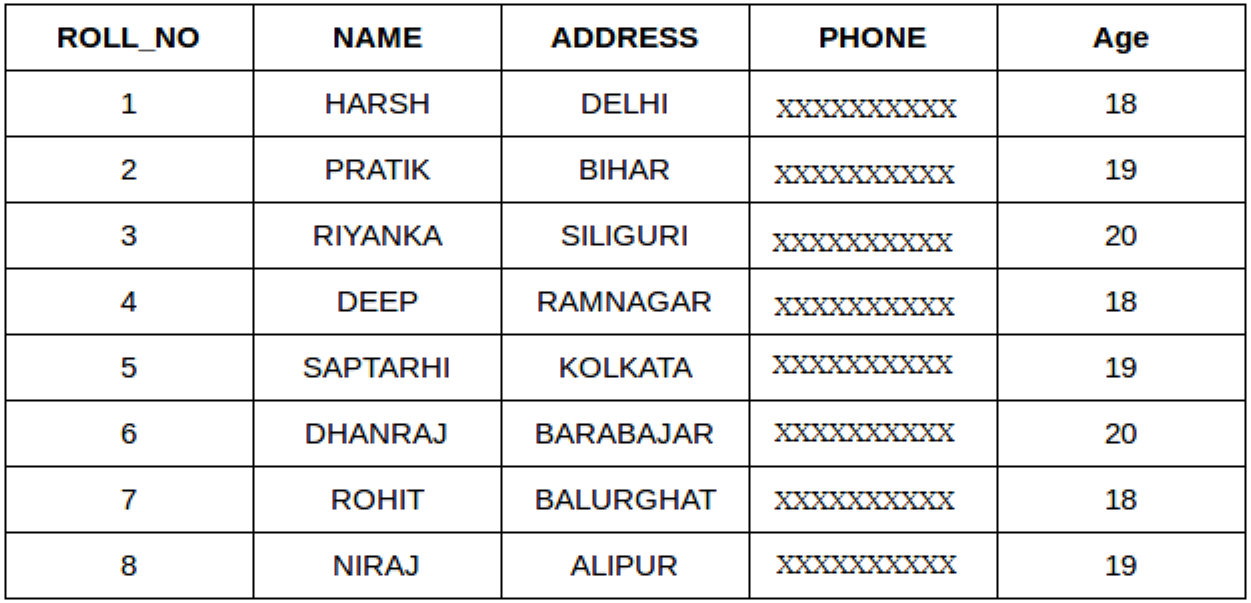
Select all (or Project) the attributes of the EMPLOYEE table at site 1 and then transfer them to site 3.
For this, we will transfer NAME, DID(EMPLOYEE) and the size is : **30 * 1000 = 30000 bytes.**

Transfer the table DEPARTMENT to site 3 and join the projected attributes of EMPLOYEE with this table. The size of the DEPARTMENT table is : **30 * 50 = 1500 bytes.**

Applying the above scheme, the amount of data transferred to execute the query will be : **30000 + 1500 = 31500 bytes.**

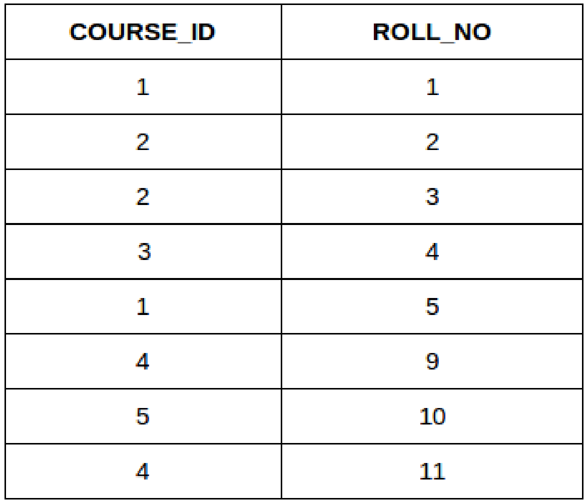
SQL | Join

Student



ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

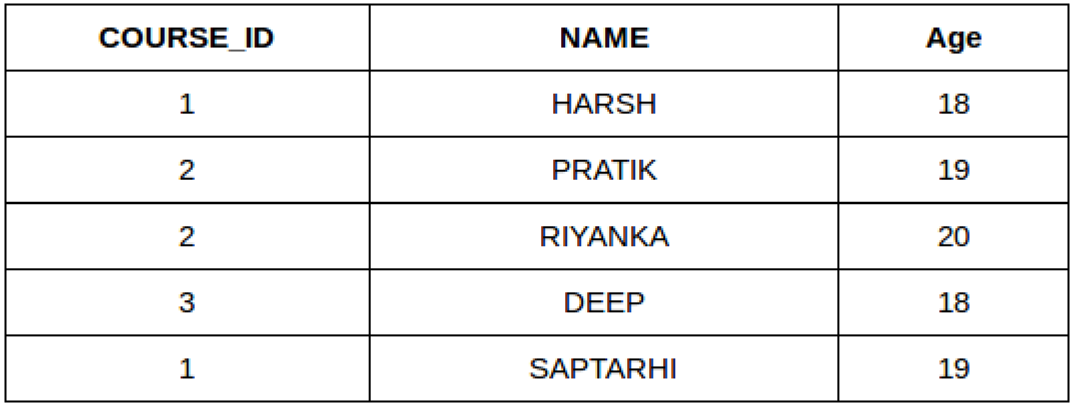
Student Course



COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

SQL | Join

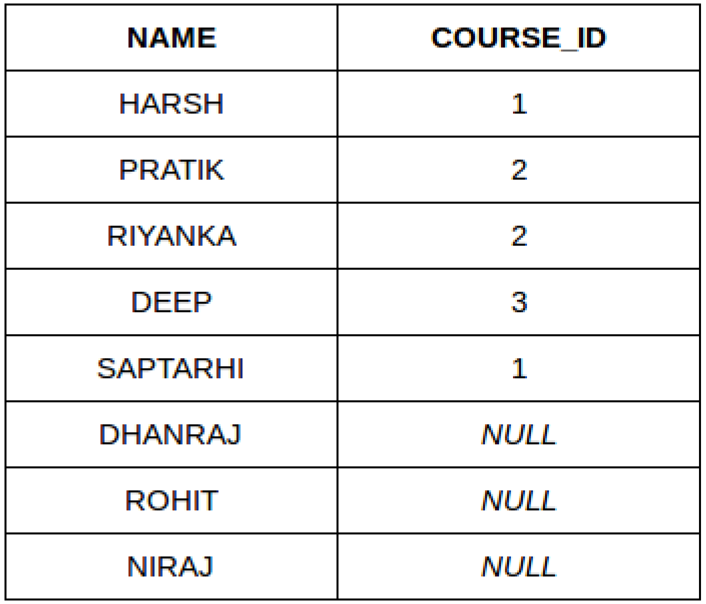
```
SELECT StudentCourse.COURSE_ID,
Student.NAME, Student.AGE FROM Student
INNER JOIN StudentCourse ON
Student.ROLL_NO = StudentCourse.ROLL_NO;
```



COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

SQL | Join

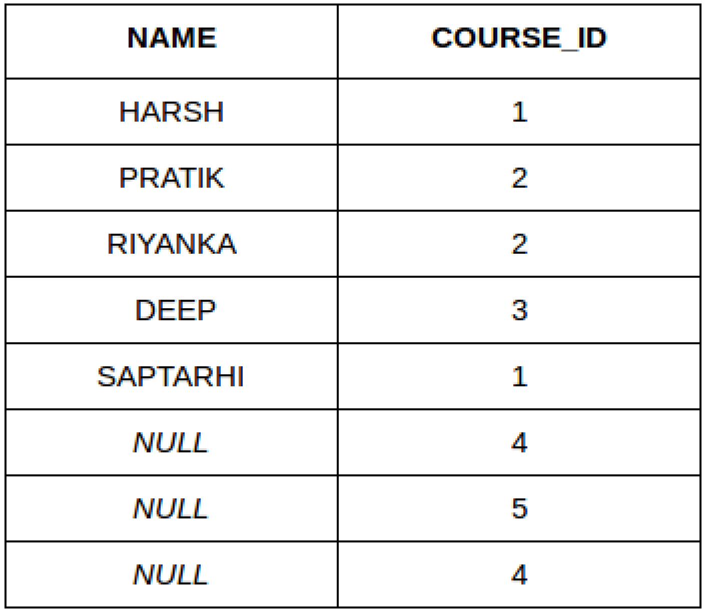
```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student LEFT JOIN StudentCourse ON
StudentCourse.ROLL_NO = Student.ROLL_NO;
```



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

SQL | Join

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student RIGHT JOIN StudentCourse ON
StudentCourse.ROLL_NO = Student.ROLL_NO;
```



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

SQL | Join

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student FULL JOIN StudentCourse ON
StudentCourse.ROLL_NO = Student.ROLL_NO;
```

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	NULL
ROHIT	NULL
NIRAJ	NULL
NULL	4
NULL	5
NULL	4